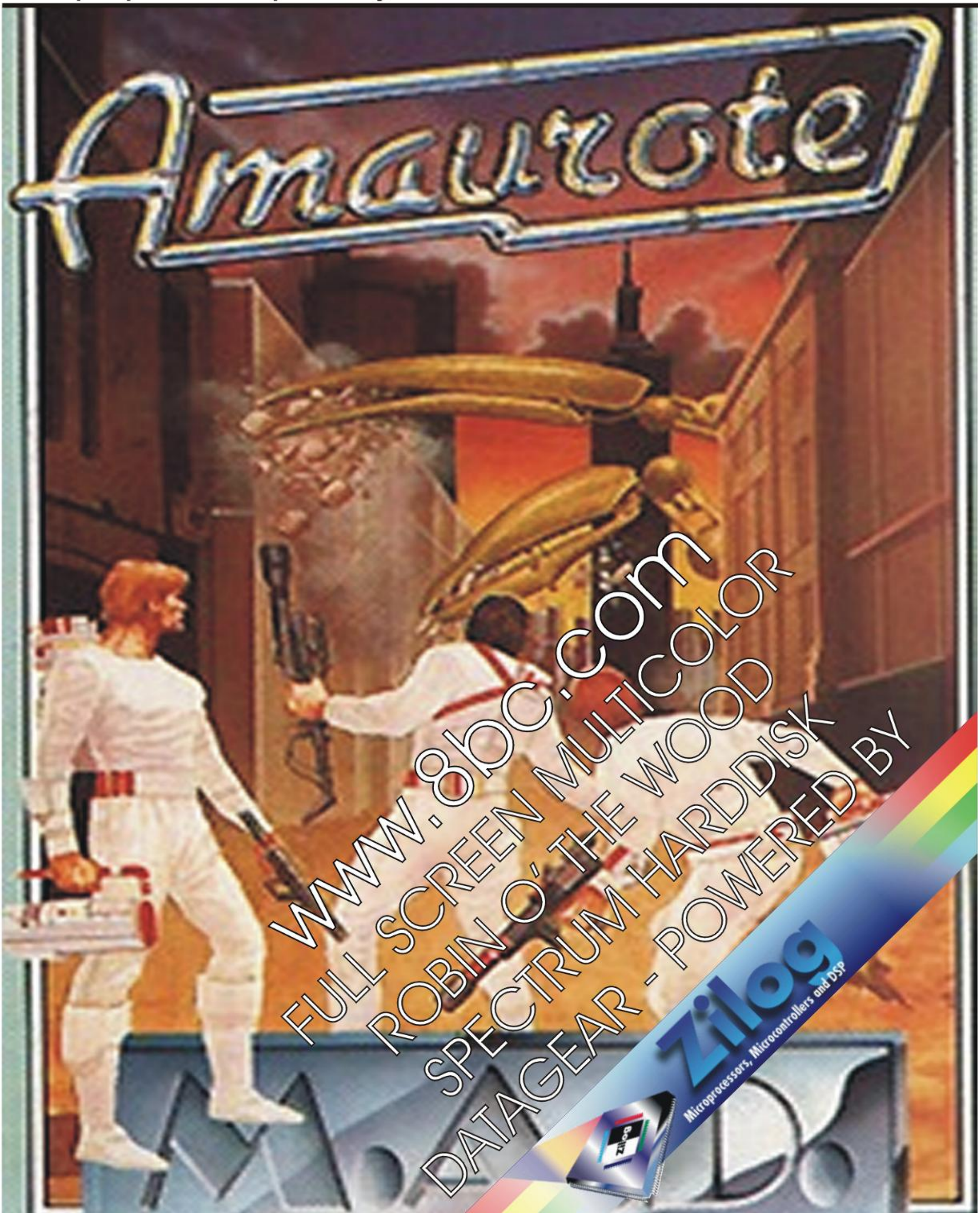
Your Spectrum

Časopis pravého Spectristy

YS #8-9: červen '99



YOUR SPECTRUM 08-09/99
časopis určený výhradně pro uživatele
počítačů ZX Spectrum a kompatibilních

Distribuce, předplatné:
8BitCompany Publishing
Tomáš Modroczki
Pražská 2532
438 01 Žatec
Česká republika
tel.: 0602/472579

Adresa redakce:
8BitCompany
Martin Blažek
Luční 4570
760 05 Zlín
Česká republika
tel.: 0603/543256

Internet: www.8bc.com
e-mail: 8bc@publikum.cz

Redakční rada:
Martin Blažek-Blažko/systems **-BLS-**
Jan Kučera-Last Monster **-LMN-**
Tomáš Modroczki-A. I. D. S. **-AIDS-**
Slavomír Lábsky-Busysoft **-BUSY-**

Obsah YS 08-09/99:

Toto číslo je věnováno všem.





Milí Spectristé!

Někdy je až neuvěřitelné, jak to letí. Jako by to bylo včera, co vyšlo YS 07/98. Po nahlédnutí do kalendáře jsem s hrůzou zjistil, že od vydání posledního čísla YS uplynula delší doba než malá...
A tak velká omluva všem poctivým čekatelům na nové YS především jmenovitě ode mne. Spousta jiných povinností mne donutily se dočasně věnovat také něčemu jinému než kompilování nového časopisu. Ostatně pěkně o tomto problému píše Tritol v novém (též maxičísle) časopisu ZX Magazín (gratuluji Tritole, sám bych to líp nenapsal). Zkrátka, budeme se snažit, aby k takovýmto výlukám nedocházelo, ale znáte to, člověk míní a Pánbů...
Ale k věci: toto číslo jsme se snažili opavdu nabušit-schéma DataGearu, recenze klasických ale výborných her, výpis full-screen Multicolor rutiny, konečně definitivní podoba SFS-hard-diskového formátu pro ZXS, rozhovor s Markem Jonesem... zkrátka věřím, že vám toto číslo YS vydrží minimálně do vydání čísla nového ;-)
A ještě jedna věc. Máte-li přístup k Internetu, zkuste někdy nahlédnout na **www.8bc.com** do oddělení ZXS. Časem tam najdete spoustu věcí a navíc se zde můžete zaregistrovat coby aktivní uživatelé ZXS. K tomuto registru pak budou mít všichni přístup a tak budou vědět, kdo co dělá.
A kdo co dělá? Tritol vyvíjí MBC, tedy MB-Commander, konečně komfortní software pro práci se soubory na systému MB-02+, paralelně s tím se v KpSH (Komise pro standardizaci harddisku) horentně pracuje na návrhu hardwaru harddisku, AIDS vyvíjí další a dokonalejší a lepší a rychlejší a... D40/D80 Emulátor pro MB-02+... BUSY se přestěhoval do Otrokovic u Zlína, aby kvůli vývoji ve věci SFS byl blíže 8BC.
Ode mne je to vše, pište, pište, pište, vaše články opravdu otiskneme. A **díky** za věrnost!

-BLS-

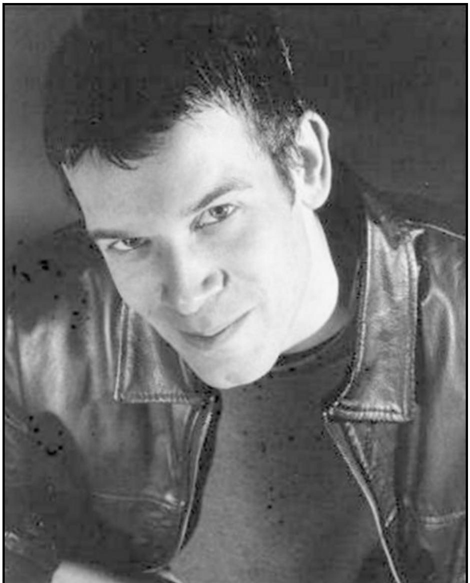
KUKADLO DO SVĚTA

Rozhovor s Markem Jonesem

2. část

V minulém čísle YS jsme se seznámili s jedním z otců klasických her pro ZXS Markem Jonesem. Dnes se dozvíte více o jeho kariéře.

YS: Na které spektrácké hře jsi se podílel a jak?
MJ: První, co jsem dělal bylo, že jsem upravil nahrávací obrazovku hry Road Race, což byla mizerná hra se závodničkama, která se zdarma přikládala k nějakému časopisu (tuším to byl Your Sinclair). Dělal jsem logo hry a logo Oceanu. Udělal jsem se ovšem na Wizballu, kde jsem tvořil veškerou grafiku. Dělal jsem hudbu pro Arkanoid ve Whamu (a dostal tak Ocean do průšvihu s Melbourne House!). Dále jsem dělal grafiku pro Gryzor, The Vindicator a něco do Dragon Ninji (mimochodem, některé vlaky mají z boku napsáno BTF. To byla zkratka pro "Bugged to Fuck" (něco jako "zasrá*o chybami), opravdu jsme měli pocit, že programátor byl strašný!). Dělal jsem nahrávací obrazovky pro hry MagMax, Wizball, Gryzor, Mutants, Tai-Pan, The Vindicator, Arkanoid II a Dragon Ninja. Vypomáhal jsem na designu hry Platoon, díl v tunelech byl můj nápad-byl jsem doma u Simona Butlerse a vzpomínám si, jak jsem mu to navrhl, ačkoliv ten by teď určitě oponoval, že to byl jeho nápad (Simone, jestli tohle čteš (a umíš česky-pozn. editora), mám fakt dobrou paměť a přesně si vzpomínám na ten okamžik, kdy jsem to řekl!). Trochu jsem taky pracoval na Total Recallu a jediný ždibec, který se z mé práce ve hře použil, byla úvodní obrazovka, která je vidět mj. i ve hře, pokud si počkáte. Mimochodem, do většiny mých úvodních obrazovek jsem psal spoustu blábolů černým inkoustem na černý papír (ty jsou vidět jen když zrušíte barvy). Ani si už nepamatuji, do kterých obrazovek jsem psal co, ale je-li někde nevyužité místo, máte velkou šanci najít nějakou blbost, zrušíte-li atributy (pokud to zkusíte a něco najdete, budu rád, když mi namailujete černobílý obrázek, abych to taky viděl).
YS: Jaká je tvá oblíbená spektrácká hra?
MJ: Tak teď nevím, buďto Tir Na Nog nebo Legend of Avalon.
YS: A jaká je tvá nejlepší a nejhorší vzpomínka na ZXS?
MJ: Nejlepší: když jsem si sehnal novou hru a byl schopen se do ní ponořit tak, jako bych tam byl (teď už to neumím!). Nejhorší? R Tape loading error.



Ďábelský výraz ve tváři Marka Jonese svědčí o tom, že kdysi dávno byl pravým Spectristou.

YS: Řekni nám něco o Oceanu...
MJ: Sídlo bylo v suterénu starého kostela na Central St. v Manchesteru (přezdívalo se tam tomu "Žaláře"). Byl tak klídek, možná až příliš velká pohoda, páteční večeře oficiálně trvaly 1 hodinu, neoficiálně 3 hodiny, protože jsme to všechno měli zaplacené. K 17. narozeninám jsem dostal Strip-O-Gram (patrně dárek spočívající ve velkém dortu, ze kterého se vylíhne nahá slečna), byli tam všichni včetně bossů, tehdy jsem se cítil fakt trapně. Byla tam taky podlouhlá místnost, říkali jsme jí Arcade Alley ("Pařanská alej"), ve které byly všechny mašiny. Vánoční večírky byly vždycky skvělé se všemi těmi lidmi, kteří dělali spousty legrace aj. Už nikdy

nezapomenu na Davida Warda jak tehdy házel nohama na "New York, New York"! Dělala tam hromada výborných lidí: Simon Buttler, Martin MacDonald, Dawn Drake, John Meegan, Kane Valentine, Rocky Ming, Lorraine Broxton, Steve Wahid, Jonathan Dunn, Martin Galway, Mike Lamb, Lee Cowley, Gary Bracey, s holka z patra vždycky byla sranda, Paul Hughes a spousta jiných, na které si teď nemůžu vzpomenout. Po pravdě musím říct, že s některými jsem se tak spřátelil, že i rok poté, co jsem odešel, jsem se vrátil na vánoční party.

YS: A co děláš teď?

MJ: Dělám v obchodě s nezávislou muzikou, jsem DJem ve třech indies/popových nočních klubech a manažerem skupiny Glendon, mají internetovou stránku www.vegetableman.demon .co.uk.

YS: Chtěl bys něco vzkázat Spectristům?

MJ: Hlavně to všechno udržujte na živu, protože jsem se trochu obával, že by to všechno mohlo zmizet (hry a to všechno). S příchodem Internetu mohou všechny ty věci bez problémů být tady všude mezi námi po tisíce let dopředu. Dlouho poté, co se originální kazety a časopisy rozpadnou. Je totiž skvělé předstírat, že je ti zase 14!!!

-BLS-

REFLEXE OD JINUD...

Jednoho jarního dne si takhle listuji Reflexem 19/99 a co to nevidím: přes celou stránku se rozpíná článek "SPECTRUM Á LA LINUX?" o siru Sinclairovi a jeho ratolestech. Rozběhl jsem se jako amokem posedlý a utíkal jej ukázat LMN. Oba jsme se pak zuřivě dali do čtení. Pokud se vám nepodaří jmenovaný Reflex sehnat, celý článek vám (bez laskavého svolení Reflexu) přinášíme.

~

Legendární anglický vynálezce Clive Sinclair se vrací. Tvrdí: dokážu vyrobit plně funkční přenosný počítač, který bude stát o polovinu míň než dnešní srovnatelná pécéčka. Proč bychom něco takového měli věřit chlapíkovi, který v počítačové branži už více než patnáct let nepracuje? Protože se mu podobný husarský kousek podařil už počátkem osmdesátých let. Malými počítači značky Sinclair zaplavil trh a ovlivnil generaci dnešních třicátníků, kteří v devadesátých letech rozhýbali informační revoluci.

~

Nevíte, co je to Spectrum? Pak jsou jen tři možnosti: a) jste dvacetiletý cucák, který přišel k počítačům až ve chvíli, kdy vzduchem létaly megahertzy a megabyty, b) o počítače jste se nikdy nezajímali a dodnes vám tak trochu lezou na nervy, c) máte opravdu špatnou paměť.



Sir Clive Sinclair a jedna z mnoha slavných fotografií. Zde ve svých rukou třímá příruční Z88.

Malé počítače ZX 80, ZX 81 a Spectrum byly ve své době fenoménem, který nemá obdoby. Připojovaly se k televizi, šlo na nich spouštět programy, a hlavně hrát hry. Byly dokonale slepou vývojovou větví (tuplovaná hloupost!- pozn. editora), ale přesto znamenaly mnoho. Staly se prvními počítači, které lidem vlezly až do obýváku. A naučily je, že komputery nejsou jen studené bzučící obludy (dobré tak akorát do sci-fi filmů), ale že se mohou stát běžnou spotřební věcí. Nejen docela užitečnou, ale taky zábavnou.

Sinclairovy počítače patřily v první polovině osmdesátých let mezi vyhledávané klukovské poklady-a to všude na světě. Do Čech se dovážely z ciziny a měly stejný glanc jako třeba originální levisky. Později se začaly vyrábět také tuzemské obdoby (třeba nezapomenutelně ošklivý počítač IQ 151), ale to už bylo v době, kdy se ve světě pomalu prosazovala pécéčka. Malé osmibitové počítače jsou už dnes historií (či možná spíš prehistorií), ale pro pamětníky opentlenou notnou dávkou sentimentu. Hodně lidí z branže se právě na nich naučilo programovat**.**Pozoruhodnou figurou je i tvůrce Spectra Clive Sinclair, kterému v Británii přezdívají "Strýček Clive". Nepatří mezi vizionáře typu Billa Gatese, Steva Jobse či Andyho Grovea. Není typickým počítačovým podnikatelem. Spíš je šmrncnutý osobností českého Járy Cimrmana-až na to, že jeho vynálezy jsou skutečné. "Strýček Clive" začal v Británii jako první vyrábět či prodávat kapesní kalkulačky a digitální hodinky, ale například také neúspěšně experimentoval s auty 7na elektrický pohon. Vždy měl víc nápadů, než dokázal realizovat. Celý život: narodil se v anglickém Surrey v roce 1940, jeho otec i dědeček byli inženýři. Otec navíc podnikal, takže Clive už v dětství poznal, co to znamená vydělat na nějakém nápadu spoustu peněz, ale také přijít velmi rychle o všechno. Už ve škole ho bavila matematika a fyzika, sestrojil si vlastní počítačku. Převáděla čísla na jedničky a nuly, což považoval za svůj objev. *"Byl jsem velmi rozladěný, když jsem zjistil, že binární soustava se běžně používá. Doufal jsem, že na tom nápadu zbohatnu."*

Místo vysoké školy nastoupil jako redaktor do časopisu pro radioamatéry. Počátkem šedesátých let si založil vlastní firmu, která podnikala hlavně v elektronice. Koncem sedmdesátých let si Sinclair přečetl ve Financial Times článek, ve kterém se předpovídalo, že se do pěti let začnou prodávat malé počítače v ceně kolem sto liber. Okamžitě cítil, že to je oblast, kde by se mohl prosadit. Ale netrvalo to roky, nýbrž několik měsíců: v lednu 1980 představil počítač ZX 80, který stál 99,95 libry (a jako stavebnici si ho bylo možné pořídit dokonce za 79 liber). Byl to experiment, ale odezva trhu byla fenomenální. Firma byla během několika dní zavalena objednávkami. Během roku navíc Sinclair představil počítač na výstavě v Las Vegas, takže se začal prodávat i v Americe. Konkurence prakticky neexistovala. V březnu 1981 měl premiéru ZX 81, kterého se během roku prodalo téměř čtvrt miliónu kusů. A v dubnu 1982 přišlo Spectrum: počítač vybavený slušnou grafikou (s barvami) a zvukovým generátorem.

Malé počítače sice v průběhu osmdesátých let bezpečně převálcovaly pécéčka, ale Clive Sinclair ze scény nezmizel. Dál experimentoval a vynalézal, třebaže už mimo počítačovou branži. A když se o něm zrovna nepsalo v magazínech o byznyse, zaměstnával coby excentrik bulvární listy. V roce 1983 dostal Sinclair šlechtický titul, ale revolverové žurnalisty spíš zaměstnával (a dodnes zaměstnává) jeho bouřlivý milostný život-a hlavně avantýry s mladými dívkami.

Tohle všechno jsou důvody, proč má Clive Sinclair zajištěnou stálou mediální pozornost. Tím spíš, že říká, co lidi chtějí slyšet. *"Běžné pécéčko je drahé kvůli procesoru od Intelu a softwaru od Microsoftu, který vyžaduje příliš mnoho paměti,"* prohlásil minulý týden. *"Proč vyrobím počítač levnější? Bude mít míň paměti, procesor nižší třídy, jednodušší zdroj energie a levnější operační systém."*

Zatím není důvod vyrážet do papírnictví či knihkupectví (tam se v osmdesátých letech v Británii Spectra prodávala), protože Sinclairův počítač nebude k mání dřív než za dva roky. Kritici poukazují na fakt, že Sinclairova firma nemá ani vlastní webovou stránku či e-mailovou adresu (není pravda-s Clivem se můžete spojit na stránce www.sinclair-research.co.uk; pozn. editora). Ale to by nemuselo vadit: Sinclair už jednou z pozice outsidra zamával celým počítačovým světem a fanoušků má-nejen na Internetu-stále dost.

-MILOŠ ČERMÁK-

A abych to všechno ještě doplnil, uvádním kontakt na Sinclair Research:

Sinclair Research Ltd
The Penthouse
7 York Central
70 York Way
London N1 9AG
Velká Británie

Telefonní číslo: +44 171 8376150
Faxové číslo: +44 171 2783101
Internet: www.sinclair-research.co.uk

-BLS-

GAMESY

AMAUROTE

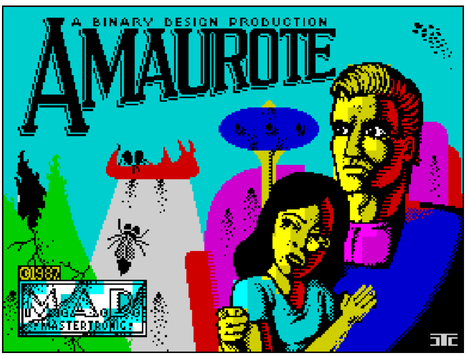
© 1987 Mastertronics

Written and designed by John Pickford
Graphic by Ste Pickford
128K Music by David Whittaker
A Binary Design Production



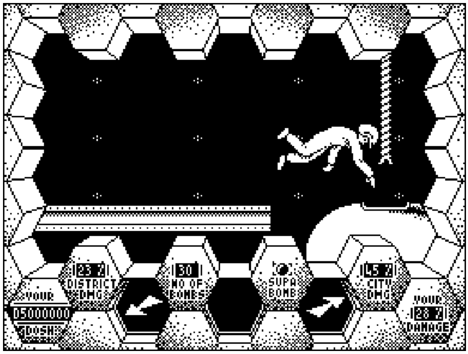
Průběh hry

Ke splnění úkolu dostáváš výbavu: Arachnus 4 (vyzbrojený chodící modul), radar, komunikační rádio, 30 odrazových bomb a hotovost $5.000.000 k nákupu nové výbavy. K dispozici máš také mapu.



Město

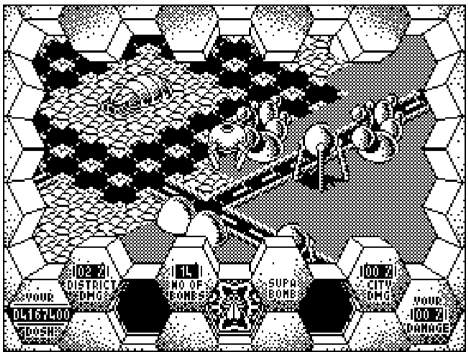
Kolonie můžeš likvidovat v jakémkoliv pořadí. Každý distrikt města Amaurote je na mapě reprezentován malým kotoučem, jehož název se objeví v horní části obrazovky jakmile na něj vstoupíš tvým modulem. Vyber si oblast, kterou začneš a stiskni FIRE. Jakmile zničíš veškerý hmyz ve zvolené oblasti, vracíš se do mapy a můžeš pokračovat dále.



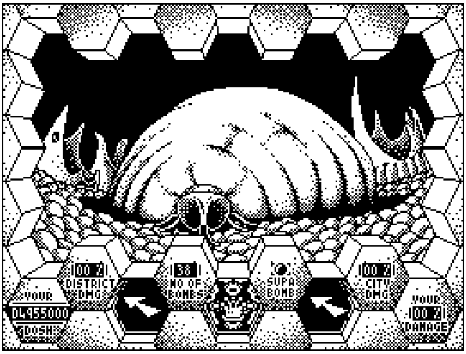
Ukázka části animace z opravdu našlapané verze Amaurote 128.

Hmyz

Ve hře se vyskytují 3 druhy hmyzu, všechny stejně ničemné a drzé.

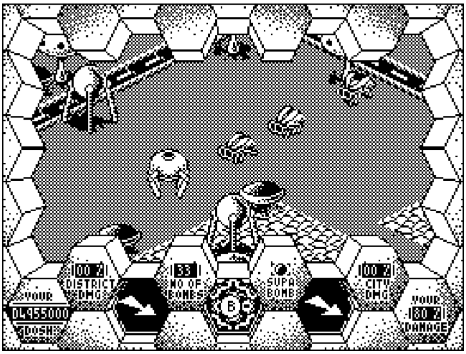


Královna: vzhledem k tomu, že je v každém distriktu tím nejdůležitějším, je také tvým hlavním cílem. Napřed se k ní ale musíš dostat - cestu ti bude komplikovat kolem lezoucí potvory. Královna instruuje ostatní hmyz, jak se chovat a navíc produkuje nový; jakmile se ti podaří zničit jednoho z protivníků, zakrátko je na světě další.



Královna těsně před "plastickou operací". Jak to vypadá po ní? Zahrajte si Amaurote 128!

Skautíci: tak tihle špehové létají kolem hledajíc potravu a vetřelce, čili tebe! Pokud tě vyšpehují, prásknou tě královně a to je průšvih!



Zástup mumlačů v závěsu. Chtělo by to tři dobře mířené...

Mumlači: na tyhlety narazíš nejčastěji. Jsou natvrdlí a taky se nikdy nevzdávají. Jejich úkolem je jednak sběr potravy pro královnu, jednak pak ochrana kolonie před vetřelci.

Bomby

Tvé zásoby skákacích bomb jsou omezeny a proto radím-šetři. Stiskem "fire" bombu vystřelíš ve směru tvého posledního kroku. Bomba se "vydá" příslušným směrem dokud nezlikviduje hmyz nebo nenarazí do budovy (měj ovšem na paměti, že tvým úkolem není zdemolovat město!).
Skákavou bombou nelze zlikvidvat plot po obvodu města a ani samotná královna. Pokud se chceš zbavit té, potřebuješ "supa-bombu". O tu si ovšem musíš říci prostřednictvím rádia. Jsou drahé a nebezpečné, takže bacha!
Jakmile vypálíš bombu, nemůžeš vypálit další, dokud ta první neexploduje (bez ohledu na to, zda zničí cíl, či ne). V mezičase bude červeně problikávat ukazatel počtu zbývajících bomb.

Rádio

Od svého velícího důstojníka jsi vyfasoval vysílačku. Pokud ji zapneš, objeví se následující možnosti:

1. SUPA BOMB (žádost o "supa-bombu")
2. REQUEST BOMBS (žádost o bomby)
3. RESCUE (quit-zbabělý úprk)
4. REPAIR (žádost o servis)

Bomby jsou spuštěny přímo do města, použijte scanner a najdete je.
Je-li tvůj Arachnus 4 příliš poškozen, můžeš zakoupit nový; snažte se ovšem zbytečně neplýtvat penězi.

Scanner

Pomocí scanneru můžete lokalizovat nejbližší hmyz, královnu a bomby. Zadejte co hledáte a následujte šipky.

Ovládání

Vpravo vzhůru	Y - P
Vlevo vzhůru	Q - T
Vpravo dolů	H - L
Vlevo dolů	A - G
Střelba	B - M, Space
Rádio	Caps Shift
Změna barev	V
Scanner-hledej hmyz	Z
Scanner-hledej královnu	X
Scanner-hledej bomby	C

-BLS-

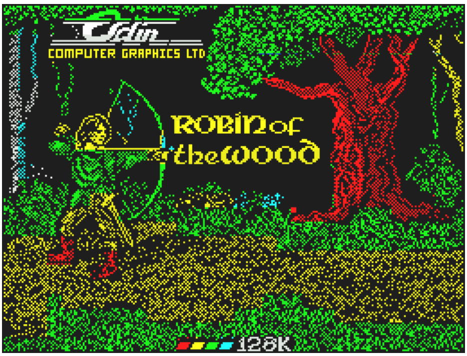
Nápad:	■■■■■■■□□
Hratelnost:	■■■■■■□□□
Grafika:	■■■■■■■■□
Zvuk:	■■■■■■■□□
Verdikt:	40. číslo herního časopisu pro ZXS Crash dalo v květnu 1987 hře Amaurote hodnocení 92% ("Úžasné dílo... velmi hratelné a návykové."). Co dodat?

Robin o' the Wood

© 1985 Odin

Tak kvůli téhleté hře jsem se na základce házel marod, jen abych mohl sedět doma u Spektráče a hrát... a hrát... a hrát. Opravdu krásný kousek: nádherná barevná grafika, hezká hudba, dobrá hratelnost a špičková atmosféra. Akorát se mi nikdy nepodařilo tuto hru projet...

...až teď po 12ti letech jsem sedl a za dva dny hru konečně dojel. Nyní vám poradím, jak na to. Po nahrání si zvolte ovládací prvky a spusťte hru. Objevíte se buďto v lese a nebo v žaláři; v tom druhém případě musíte najít klíč, díky kterémuž se dostanete ven do sherwoodského lesa. Dobrod-ružství může začít.



Robin o' the Wood existuje také ve verzi pro ZXS 128 (vyšla v roce 1986). Oproti původní 48K verzi jsou zde nepatrné změny.

Napřed se podíváme na to, co v lese můžete najít:



Lesní kvítek. Sám o sobě je vám k ničemu. Najděte lesní vílu (ta se zjevuje na konkrétních místech v lese) a začnou se dít kouzla.



Toto je volně se povalující život. Určitě berte.



Šípy jsou moc fajn, ale jsou vám na nic, když nemáte luk. Jak jej získat se dozvíte níže.



Měšec s penězi. Ten nezískáte jen tak snadno-musíte oloupit kněze. Najděte jej a zlikvidujte stráže za ním-kněz vám ze strachu vydá 2 měšce a uteče.



Klíč se vyskytuje pouze na dvou místech: v žaláři (abyste se dostali ven do lesa) a u Sherwoodského hradu (abyste se dostali dovnitř).

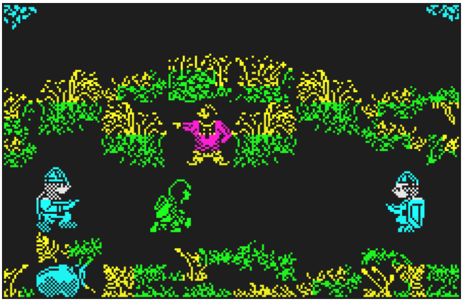
V lese se pohybuje také spousta jiných osob a přízraků. Všechny strážce likvidujte tím, co zrovna máte (palice, meč, luk s šípy). Divoké prasata nezničíte, snaže se jim vyhnout.



Právě se vám zjevila lesní víla. Nemáte-li u sebe nic, nezjeví se. Máte-li u sebe pouze měšce s penězi, jeden vám zabaví. Máte-li ovšem u sebe lesní květiny, stane se následující: jeden kvítek vám víla zabaví a nedostanete nic. Za dva kvítky vám ovšem doplní energii na maximum a za tři vás přenese k sherwoodskému hradu.



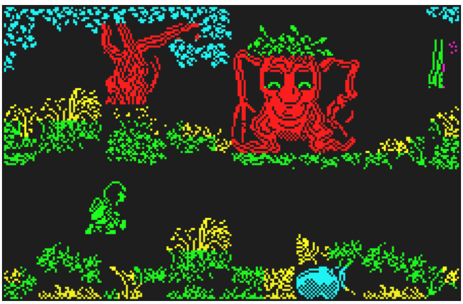
Kouzelný děda. Má-li dobrou náladu, doplní život a energii.



Dostali jste se do šerifovy léčky. Z té se nedostanete zavřou vás do žaláře (z toho samozřejmě utečete naděte klíč a hurá do sherwoodského lesa).



Konečně! Podařilo se vám najít kněze i s ochrankou. Zbavte se stráže a dva měchy plné penízků jsou vaše.



Kouzelný strom vyhledejte, máte-li u sebe alespoň 3 měšce s penězi (přesně ty, o které oloupíte kněze). Za každé 3 měšce postupně dostanete: meč (budete používat místo palice), luk (funguje pouze když máte u sebe šípy), 1. sherwoodský šíp, 2. sherwoodský šíp a 3. sherwoodský šíp (neplést s klasickými šípy do luku!-sherwoodské šípy nejsou určeny na střílení). Až dostanete všechny 3 sherwoodské šípy, posbírejte 3 lesní kvítka, vyhledejte vílu a nechejte se přenést k sherwoodskému hradu. Vběhněte dovnitř a vyhledejte nově otevřený vchod...

COMPUTER GRAPHICS LTD.

-BLS-

Nápad:	■■■■■■□□□
Hratelnost:	■■■■■■■■□
Grafika:	■■■■■■■■□
Zvuk:	■■■■■■□□□
Verdikt:	Krásná barevná grafika, fajn atmosféra, hýbe se to rychle a dá se to projet i bez pouků. Klasická hra, která by žádnému Spectristovi neměla chybět.

PROGRAMOVÁNÍ

Zázraky v BASICu

díl 06-Song in Lines

Pokiaľ máte radi všeliaké grafické efekty potom je tu pre vás nasledujúci program. Je to ukážka hlavného algoritmu kreslenia čiar použitého v známej sérii programov "SONG IN LINES". Program je napísaný pre počítač ZX Spectrum, ale dá sa bez problémov prepísať aj na iné počítače.

```
10 REM Busysoft
20 OVER 0: INVERSE 0
30 INPUT "Pocet ciar:";c: LET x=23677:
LET y=23678
40 INPUT "Mazanie ciar pomocou INVERSE
???(1=ano/0=nie)";m
50 RESTORE : DIM s(c,4): DIM d(3,4): IF
NOT m THEN OVER 1
60 FOR a=1 TO 3: FOR b=1 TO 4: READ
d(a,b): NEXT b: NEXT a
70 CLS : DATA 50,10,10,50,3,4,3,2,
```

```
255,175,255,175
80 FOR a=1 TO c: FOR b=1 TO 4: LET
s(a,b)=0: NEXT b: NEXT a
90 FOR a=1 TO c: INVERSE m: GO SUB 150:
FOR b=1 TO 4
100 LET d(1,b)=d(1,b)+d(2,b)
110 IF d(1,b)>0 AND d(1,b)<d(3,b) THEN
GO TO 130
120 LET d(2,b)=-d(2,b): LET
d(1,b)=d(1,b)+2*d(2,b)
130 LET s(a,b)=d(1,b): NEXT b
140 INVERSE 0: GO SUB 150: NEXT a: GO
TO 90
150 PLOT s(a,1),s(a,2): DRAW s(a,3)-
PEEK x,s(a,4)-PEEK y
160 PLOT d(3,1)-s(a,1),s(a,2): DRAW
d(3,3)-s(a,3)-PEEK x,s(a,4)-PEEK y
170 PLOT d(3,1)-s(a,1),d(3,2)-s(a,2):
DRAW d(3,3)-s(a,3)-PEEK x,d(3,4)-
s(a,4)-PEEK y
180 PLOT s(a,1),d(3,2)-s(a,2): DRAW
s(a,3)-PEEK x,d(3,4)-s(a,4)-PEEK y
190 RETURN
```

Program sám o sebe ide pomerne pomaly. Je to dané najmä tým že veľmi veľkú časť celého času zaberajú rôzne výpočty a ošetrenia slučiek FOR NEXT. Komu sa chce, môže si tento program skompilovať. Pozor však na príkaz DIM s(c,4) na riadku 50. Niektoré kompilátory (našťastie nie všetky) totiž vyžadujú vopred poznať rozmery dimenzovaných premenných. V tom prípade dosaďte namiesto premennej "c" do tohto príkazu nejaké dostatočne veľké číslo (napr. 200). Potom by ste ale nemali pri spúšťaní skompilovaného programu zadávať väčší počet čiar ako toto číslo.

Program využíva výhradne celočíselnú aritmetiku, preto môžete použiť aj celočíselný kompilátor. Kto si trúfa, môže samozrejme program prepísať priamo do strojového kódu. Vďaka celočíselnej aritmetike to nebude ani moc ťažké. Tento program nevyužíva žiadne neštandartné alebo nedokumentované funkcie a služby a ani sa nepotrebuje synchronizovať so snímkovou frekvenciou monitora, preto by mal správne fungovať na všetkých počítačoch a emulátoroch ktoré sú kompatibilné s počítačom ZX Spektrum na úrovni basicu. Pár poznámok k prenositeľnosti na iné počítače: Program je písaný pomerne jednoducho, takže prepísanie do iného basicu by nemal byť problém. Čísla 255 a 175 na riadku 70 udávajú počet bodov grafického režimu zmenšený o 1 (255 po osi X, 175 po osi Y). Príkaz PLOT X,Y presúva kresliaci kurzor na pozíciu X,Y a príkaz DRAW X,Y kreslí čiaru z aktuálnej pozície kurzora na pozíciu ktorá je od aktuálnej posunutá o X,Y (jedná sa o relatívne kreslenie čiary). Členy "- PEEK x" a "- PEEK y" slúžia na prepočet relatívneho posunutia pozície na absolútne súradnice tejto pozície. "x" je adresa v pamäti, kde sa uchováva X-súradnica aktuálnej pozície kurzora, podobne "y" je adresa kde je uložená Y-súradnica. Ak máte k dispozicii príkaz na kreslenie čiary pracujúci s absolútnymi súradnicami, potom tieto členy môžete vynechať. Verím že sa vám program bude páčiť a že ho prípadne použijete ako pekný grafický doplnok do nejakého vlastného väčšieho programu.

-BUSY-

MultiTech... jak na to?

lekce 03: ...a kudy voda teče...

V předchozím čísle seriálu Multitech... jak na to? jsme si detailně popsali principy zobrazování na ZXS. Řekli jsme si, kde jsou kameny úrazu při vykreslování barev. Víme, že zatím neumíme pokrýt celou plochu obrazovky režimem Multicolor za použití procesoru Z80-CPU, zvládneme to ovšem za asistence čipu Z80-DMA.

Jak jistě víte, Z80-DMA je součástí MB-02+. Pro ty méně šťastné, kteří MB-02+ nevlastní budiž alternativou řešení jménem DataGear. O tom se dočtete v tomto vydání YS, navíc si jej podle našeho schématu můžete postavit sami. Pak již nebude stát nic v cestě dnešnímu experimentu.

```
        org 45500
ODKUD   equ 54784
KAM     equ 22528
LEN     equ 31
HADR    equ ODKUD/256

        ent $
POC     di
        ld hl,41216
        ld de,41217
        ld bc,256
        ld (hl),162
        ldir
        ld a,161
        ld i,a
        im 2
        ld de,16384
        ld hl,48640
        ld bc,6144
        ldir
```

```
        ei
BREAK   halt
        call 8020
        jr c,BREAK
        ld a,63
        ld i,a
        im 1
        ret

        org 162*256+162
START   push af
        push bc
        push de
        push hl
        push ix
        ld d,#1D
        ld e,HADR
        push de
        ld hl,KAM
        push hl
        ld hl,DMA
        ld b,DELKA
        ld c,11
        otir
        ld hl,KAM+32
        ld de,ODKUD+256
        ld b,23
LOOP11  push bc
        ld (OD),de
        ld (KA),hl
        push hl
        ld hl,DMA2
        ld b,DELKA2
        otir
        pop hl
        inc d
        push de
        ld de,32
        add hl,de
        pop de
        pop bc
        djnz LOOP11
        ld hl,DMA3
        ld b,DELKA3
        otir
        ld b,198
LOOP1   djnz LOOP1
        ld bc,32*256+11
        ld hl,#87B3
        ld de,#CF0D
        ld a,b
        ld lx,24
        out (c),e
        out (11),a
LMNLOOP1 out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        add a,b
        out (c),e
        out (11),a
        out (c),d
        out (c),l
        out (c),h
        xor a
        ld e,b
        ld d,a
        pop hl
        add hl,de
        pop de
        inc e
        push de
        push hl
        ld a,#AD
        out (11),a
        out (c),l
        out (c),h
        ld a,32
        out (c),d
        out (11),a
        out (c),e
        ld hl,#87B3
        ld de,#CF0D
        dec lx
        jr nz,LMNLOOP1
END     pop hl
        pop de
        pop ix
        pop hl
        pop de
        pop bc
        pop af
        ei
        ret

DMA     defb #C3,#C7,#CB,#7D
        defw ODKUD
        defw LEN
        defb #54,#2,#50,2
        defb #C0,#AD
        defw KAM
        defb #82,#CF,#B3,#87
DELKA   equ $-DMA

DMA2    defb #1D
OD      defw 0
        defb #AD
KA      defw 0
        defb #CF,#B3,#87
DELKA2  equ $-DMA2

DMA3    defb #1D
        defw ODKUD
        defb #AD
        defw KAM
DELKA3  equ $-DMA3

DELKAC  equ $-POC
```

Vytušili jste správně: na předchozí stránce vidíte plný výpis zobrazovacího ovladače na režim full-screen Multicolor využívající Z80-DMA. Napsal jej Tomáš Modroczki (-AIDS-), Honza Kučera (-LMN-) jej posléze výrazně zkrátil a zoptimalizoval.
Spuštěním od adresy 45500 se aktivuje mód přerušení IM 2, jehož je využito k vykreslování režimu Multicolor. Na adrese 48640 je uložena obrazovka (pouze pixly bez atributů, tj. 6144 bajtů), kterou chcete Multicolorem překrýt. Na adrese 54784 následuje 6144 bajtů virtuálních atributů; ty jsou mapovány stejným způsobem jako atributy klasické, tj. 32 bajtů/1 atributový řádek, zde je ovšem atributový řádek 8x jemnější než ten klasický.
Tato rutinka je základním stavebním kamenem ke zobrazování v režimu MultiTech. V příštím čísle YS se dočtete, jak převádět obrázky fotografické kvality právě do režimu MultiTech. Ti, kteří si nehodlají zakoupit MB-02+ a ani si nebudou stavět DataGear se dočtou o tom, jak provozovat MultiTech bez těchto hardwarových peripetií.

-BLS-

Strojový kód pre pokročilých

lekcia 07

V tejto lekcii sa budeme zaoberať podobnou rutinkou ako v predchádzajúcej, avšak tentoraz číslo uložené ako postupnosť cifier v buffei nebude desiatkové, ale šestnástkové. Najprv sa pozrime na domácu úlohu z predchádzajúcej lekcie.

Jednotlivé cifry nášho čísla sme mali v pamäti uložené ako obyčajné znaky v kóde ASCII. Keď si pozorne všimneme hodnoty týchto kódov, uvidíme, že kód každej cifry je presne o 48 väčší ako hodnota, ktorú predstavuje táto cifra. Teda stačí od kódu znaku odpočítať číslo 48 (ASCII kód znaku '0') a dostaneme hodnotu cifry.
Jedným z podproblémov nášho programu bolo vynásobiť aktuálnu hodnotu čísla desiatimi. Najpohodlnejšie riešenie by bolo využiť už hotovú rutinku na vynásobenie dvoch čísel z 4. lekcie. Lenže táto rutinka násobí iba osembitové čísla. Dalo by sa to obísť tak, že by sme si začiatok rutinky upravili tak, aby jeden operand bol v celom registri DE, do registra A vložili číslo 10 a do registra HL poctivo vložili nulu (lebo D už nemusí byť nulové). Ja som vám však ponúkol principiálne iné a v danej situácii efektívnejšie riešenie. Pokiaľ násobíme hodnotu v registri dákym malým vopred známym číslom potom existuje efektívnejší spôsob, ako to spraviť. Čo keby sme skúsili spraviť toto: Register HL vynásobme dvomi (**add hl,hl**), potom si tento dvojnásobok niekam odložme (napr. do BC), po odložení tento dvojnásobok ešte dvakrát vynásobme dvomi-tým dostaneme štvornáso-bok a osemnásobok. No a na záver k tomuto osemnásobku pričítajme ten odložený dvojnáso-bok a dostávame... Ten kto hádal 10-násobok mal úplnú pravdu.
Prevod hexadecimálne zapísaného čísla na jeho hodnotu je v niečom zložitejší, ale v niečom zase jednoduchší. Kedže hexadecimálna sústava používa viac cifier ako desiatková a navyše ich ASCII kódy nenasledujú tesne za sebou (ako by sme si to želali), bude určovanie hodnoty jednotlivých cifier o niečo zložitešie. Na druhej strane treba aktuálnu hodnotu čísla násobiť šestnástimi. Ako už vieme, 16 je pekné okrúhle číslo (viď 0. lekciu) ktoré sa dá vyjadriť ako štvrtá mocnina dvojky. Preto ak chceme nejaké číslo vynásobiť šestnástimi, stačí ho len štyri krát vynásobiť dvomi. A práve v tomto je to jednoduchšie. Ešte pred tvorbou samotnej rutinky si treba ujasniť, ako máme zapísané číslice nášho čísla v bufferi (v našom príklade máme číslo 'f4b3'). Učene povedané-musíme si presne definovať vstupné údaje. Nech je naše číslo napísané ako postupnosť hexadecimálnych číslic v obvyklom poradí (vyššie rády na nižšej adrese) a zakončené ľubovoľným znakom, ktorý nezodpovedá žiadnej hexadecimálej číslici (v našom príklade je tam medzera). Hexade-cimálne číslice budeme kódovať tak, ako je to zaužívané-t. j. znakmi '0' až '9' a 'a' až 'f' (budeme používať malé písmená).
Teraz nám už nič nebráni aby sme mohli navrhnúť náš program, ktorý bude používať taký istý postup ako program z minulej 6. lekcie -
t. j. bude postupne načítavať cifry, určovať ich hodnoty a postupne počítať aktuálnu hodnotu čísla. Pri prepočte ASCII kódu znaku na hodnotu cifry sa na chvíľku zastavme. Kódy všetkých hexadecimálnych cifier neležia hneď za sebou, ako je to pri desiatkových cifrách-preto to už nemôžme mechanicky prepočítavať tak ako

predtým. Pokiaľ sa jedná o cifry '0' až '9', je všetko v poriadku. Ale čo s písmenami 'a' až 'f'? Všimnime si, že ASCII kódy týchto písmen tiež ležia bezprostredne za sebou, takže stačí od kódu písmena odčítať určitú konštatnú hodnotu a problém je vyriešený. No nie celkom. Totiž na našu smolu táto konštanta je iná ako pre číslice '0' až '9'! V tejto situácií nám neostáva nič iné ako otestovať, že o akú cifru sa jedná (číslo alebo písmeno?) a podľa toho zvoliť príslušnú konštantu, ktorú potom odčítame.

```
buffer  db 'f4b3 '      príklad nejakého čísla
hxbn    ld de,buffer    DE bude ukazovateľ do
                        buffera
        ld hl,#00 HL    bude obsahovať
                        okamžitú hodnotu čísla
hexa    ld a,(de)       prevzatie cifry z
                        buffera
        ld c,'0'        ak je jej ASCII kód
                        menši ako '0'
        cp c            tak sa už nejedná o
                        žiadnu platnú cifru
        ret c           a to znamená koniec
                        čísla a návrat
        cp '9'+1        ak sa jedná o znaky
                        '0'..'9'
        jr c,cifra      tak skok na odčítanie
                        prísl. konštanty
        ld c,'a'-10     toto je konštanta pre
                        znaky 'a'..'f'
        cp 'a'          ak je kód znaku menší
                        ako 'a'
        ret c           tak sa nejedná o cifru
                        a návrat
        cp 'f'+1        ak je kód väčší ako
                        'f'
        ret nc          tak to tiež nebude
                        žiadna cifra
cifra   sub c           odčítanie príslušnej
                        konštanty
        inc de          ukazovateľ buffera na
                        ďalší znak
        add hl,hl       vynásobenie
        add hl,hl       aktuálnej
        add hl,hl       hodnoty
        add hl,hl       šestnástimi
        ld c,a          pričítanie hodnoty
        ld b,#00        cifry k samotnej
        add hl,bc       hodnote čísla
        jr hexa         a znovu pre ďalšiu
                        cifru
```

Dnes bude domáca úloha možno trochu ťažšia ako obyčajne. Skúste sa zamyslieť nad problémom presne opačným-to znamená, že máte v registri nejaké číslo a chcete ho vypísať-čiže previesť na postupnosť číslic a túto postupnosť číslic umiestniť do nejakého buffera.

lekcia 08

V minulých lekciách sme si ukázali, ako môžeme z postupnosti číslic vypočítať hodnotu tohto čísla. Dnes si pohovoríme o opačnom probléme-ako sa dá hodnota čísla previesť na postupnosť číslic a túto postupnosť zapísať do vhodného buffera.

Keď sme menili postupnosť číslic na hodnotu, museli sme vedieť, v akej číselnej sústave je vlastne to číslo zapísané. Podobne aj pri opačnom prevode nám musí byť jasné v akej sústave bude výsledná postupnosť číslic. Pre začiatok si zvoľme ten jednoduchší prípad-že chceme dostať postupnosť desiatkových číslic.
Tento prevod je vlastne presne taká istá úloha, akú sme riešili v úvodnej nultej lekcii, keď sme chceli previesť nejaké číslo do šestnástkovej sústavy. Rozdiel je len v tom, že namiesto nás to tentoraz bude robiť počítač, ktorý má dané číslo v nejakom registri a prevádza ho do desiatkovej sústavy (namiesto delenia šestnástimi bude číslo deliť desiatimi). Kedže deliť ste sa už naučili v piatej lekcii, určite nebude problém pre vás napísať takýto program. Kto chce a trúfa si, môže si to skúsiť.
Ja vám však ponúkam iné, efektívnejšie riešenie, ktorého princíp je oveľa bližší ľudskému ponímaniu čísla. Jeho myšlenkou je postupné odčítavanie desiatkových rádov, pričom si zaznamenávame koľko krát sa nám príslušný rád podarilo odčítať. Čiže napíklad ak sa nám od čísla podarí tri razy odčítať hodnotu desaťtisíc tak vieme, že naše číslo bude mať na mieste pre desaťtisíce císlicu 3. Najvyššie možné číslo ešte zobraziteľné v párovom registri je 65535. Preto nám bude stačiť výpočet iba pre týchto päť rádov a teda aj buffer dlhý päť bajtov.
Princíp práce programu je takýto: program vezme najprv najvyšší rád-desaťtisíce a tento rád odčítava dovtedy, pokiaľ hodnota čísla neklesne pod 10000. Potom vezme najbližší nižší rád (tisíce) a od novej hodnoty odčítava pokiaľ sa to ešte dá (pokiaľ opäť hodnota neklesne tentoraz pod 1000). Tento postup vykováva pre všetky rády až po jednotkový rád.
Odpočítavanie v každom ráde sa deje nasledovne: na začatku sa vloží do akumulátora ASCII kód znaku '0'. Pri každom odpočítaní sa hodnota v akumulátore zvýši o jednotku. Keďže

ASCII kódy číselných znakov idú bezprostrene za sebou, na konci odpočítavania máme v akumulátore priamo ASCII kód číslice príslušného rádu. Tento kód sa uloží do buffera. Na ukladanie číslic je použitý register BC, ktorý obsahuje adresu pamäti kam sa má uložiť kód nasledujúcej číslice.

```
run     ld hl,54321     príklad čísla
bndc    ld bc,buffer    BC bude ukazovateľ do
                        buffera
        ld de,-10000    pre desiatkový rád
                        desaťtisíce:
        call cifra      výpočet cifry
        ld de,-1000     ...tisíce
        call cifra      výpočet tisícov
        ld de,-100      ...stovky
        call cifra      výpočet stoviek
        ld e,-10        ...desiatky
        call cifra      výpočet desiatok
        ld e,-1         ...jednotky
cifra   ld a,'0'-1      výpočet číslice
                        jedného rádu:
cif1    add hl,de       odčítavame hodnotu
                        rádu a
        inc a           zároveň počítame že
                        koľko krát,
        jr c,cif1       kým neprekročíme nulu
        sbc hl,de       oprava prekročenia
                        nuly
        ld (bc),a       uloženie kódu číslice
                        do buffera
        inc bc          ukazovateľ na ďalšie
                        miesto v bufferi
        ret             koniec a návrat
buffer  db 'xxxxx'      buffer pre vznikajúce
                        číslo
```

Keď si dobre všimnete tento program, na vaše možno veľké prekvapenie zistíte, že v slučke, kde sa odčítavajú rády (cif1) je inštrukcia sčítania! Je to tak preto, lebo inštrukcia **add hl,de** má menej bajtov ako **sbc hl,de** (ušetríme pamäť), okrem toho sa rýchlejšie vykoná (ušetríme čas) a navyše nepotrebuje mať na začiatku vynulovaný carry. Môžeme si to dovoliť podľa matema-tického pravidla, ktoré hovorí že "odčítať kladné číslo je to isté ako pričítať záporné s rovnakou absolútnou hodnotou". Preto sa na začiatku programu do registra DE vkladajú desiatkové rády ako záporné čísla. Toto naše "dolu hlavou" obrátené odčítanie má jeden zaujímavý dôsledok-príznak carry nám opačne signalizuje pretečenie. To znamená, že ak nie je pretečenie, carry bude nastavený a keď nastane pretečenie, carry bude nulový. Toto musíme zohľadniť pri teste prekročenia nuly.
Všimnime si ešte jednu vec: slučka na výpočet jednej číslice veľmi pripomína delenie odčítaním. Aj tu deliteľa (príslušný desiatkový rád) odčítavame až dovtedy, pokým delenec (hodnota čísla) nie je záporný. Kedže však potrebujeme poznať aj hodnotu zvyšku, musíme na záver slučky spraviť korekciu prekročenia nuly-presne tak, ako sme to robili pri delení v piatej lekcii. Toto posledné odčítanie, ktorým sme sa s delencom dostali až pod nulu, už nesmieme rátať do výsledku-preto sa na začiatku slučky dáva do akumulátora hodnota o jednotku menšia ako ASCII kód nuly.
Na domácu úlohu sa skúste zamyslieť nad tým, ako by vyzeral program, ktorý mení hodnotu čísla na postupnosť šestnástkových číslic.

-BUSY-

HARDWARE

Obecný popis tiskárny jako základního výstupního média

Většina uživatelů se setkává s problémem, jak nejlépe prezentovat výsledky své práce. Jako nejvhod-nější se jeví prezentace ve formě tiskových dokumentů. Tyto dokumenty jsou tištěny na různých typech tiskových strojů, počínaje jedno-jehličkovými tiskárnami a konče hlubotiskovými stroji.

Nepřehledné množství tiskových technik a tiskárenských strojů vyvolává u mnoha uživatelů otázku, která tiskárna je pro jejich potřeby ta nejvhodnější. Správná volba tiskárny však závisí na mnoha okolnostech, jako například na průměrném počtu tisknutých stran, na potřebné kvalitě tisku atd., a proto je následující text třeba brát jen jako pomoc při rozhodování.

Typové tiskárny

Typové tiskárny jsou ve většině případů zastoupeny tiskárnami s typovým kolečkem, na kterém jsou vytvořeny vzory jednotlivých tiskových znaků. Tyto tiskárny jsou ve většině případů používány jako levná alternativa psacích strojů. Podobnou technologií jsou řešeny i sálové rychlotiskárny, které tisknou najednou několik řádků a tím velice zrychlují výsledný tisk jedné strany. Vzhledem k tomu, že tisk jemné grafiky je na těchto tiskárnách technicky neproveditelný, jsou v současné době již používány velmi zřídka. Vstupní informace pro typové tiskárny zahrnují pouze indexy kódové tabulky, realizované nejčastěji typovým kolečkem nebo typovou hlavicí s vyraženými znaky na indexových pozicích. Nejčastěji je využívána kódovací tabulka ASCII, která definuje indexy všem základním tiskovým znakům latinské abecedy, číslicím, speciálním znakům (paragraf, znak dolaru atd.) a některým semigrafickým symbolům.

Jehličkové maticové tiskárny

Jehličkové tiskárny jsou založeny na principu tisku matice bodů zobrazujících jednotlivé znaky tisku. Výška matice je ve většině případů rovna násobku počtu jehliček, které tisk provádějí. Nejjednodušší jsou tiskárny jednojehličkové, které tisknou vždy jen jedinou linku tisku, zvanou mikrořádek (vzpomínáte? BT-100!-pozn. editora). Pomocí těchto mikrořádků jsou tisknuty grafické dokumenty, které jsou rozloženy (rozrastrovány) na soustavu jednotlivých mikrořádků tisku. Text je obdobným způsobem nejprve převeden na grafickou podobu, a potom tisknut jako grafický dokument.
Vícejehličkové tiskárny tisknou pomocí téhož principu, pouze s tím rozdílem, že tisknou několik mikrořádků najednou. To umožňuje kódovat textové části obdobným způsobem jako se provádí kódování u tiskáren s typovým kolečkem, a tiskárna již převádí přijatý znak na matici tiskových bodů sama. Tím také dochází ke značnému zrychlení přenosu tiskové informace komunikačním kanálem a ke zrychlení výsledného tisku. Počet jehliček vícejehličkových tiskáren je volen tak, aby tiskárna vytiskla najednou celý řádek tisku. Existují tiskárny osmijehličkové, devítijeh-ličkové atd. Se zvyšujícím se počtem jehliček roste také teoretická rozlišovací schopnost těchto tiskáren. Zatímco devítijehličkové tiskárny na jeden průchod (režim DRAFT) vytisknou dokument s rozlišením asi 60 DPI (bodů na palec, tzn. 2,4 bodu/mm), 48mi jehličkové tiskárny tisknou s rozlišením asi 360 DPI (14,2 bodu/mm). Tato hodnota je však teoretická, neboť je velmi závislá na kvalitě tiskové mechaniky a zvlášť je závislá na kvalitě barvicí pásky. Rozlišení lze zvětšit několikanásobným průchodem tiskové hlavy přes jeden řádek tisku. Takto vzniká tisk zvaný NLQ (Near Letter Quality), LQ (Letter Quality), SLQ (Super Letter Quality) a další.
Jehličkové tiskárny jsou nejpoužívanější, hlavně díky své pořizovací ceně a tím také nízké ceně jedné strany tisku. Hlavní nevýhodou jehličkových tiskáren je jejich hlučnost a rychlé opotřebování barvicích pásek, které se projevuje snižováním kontrastu tisknutých dokumentů. Při větších objemech tisku se také projeví jako nevýhoda časová náročnost tisku, která je způsobena právě principem tisku dokumentu po jednotlivých řádcích.

Tepelné tiskárny

Snaha o stálou kvalitu tisku při relativní nenáročnosti tiskáren vedla výrobce k využívání tepelné techniky tisku. Tato technologie přenáší na teplocitlivý papír pomocí bodového ohřevu obdobnou matici jako tiskárna s jehličkovou technologií. Bodový ohřev je nejčastěji realizován pomocí elektrického oblouku mezi hlavou tiskárny a papírem. Tento tisk je rychlý, stabilní a také velice tichý. Rozlišovací schopnost průměrných termotiskáren se pohybuje okolo 240 DPI (9,5 bodu/mm). Použití těchto tiskáren v běžném provozu zatím brání velice nevýhodná vysoká cena termocitlivého papíru a tím i výsledná cena tisku jednotlivé strany.
Tisk pomocí tepelných tiskáren je hromadně využíván ve faxových strojích, v příručních a kufříkových tiskárnách, a to hlavně pro svoji nenáročnost na provozní podmínky a nízkou hladinu hluku při tisku.

Inkoustové tiskárny

Inkoustové tiskárny jsou další náhradou tiskáren jehličkových. Pracují s technologií nástřiku kapiček inkoustu na normální kancelářský papír. Nejčastěji jsou koncipovány tak, aby byly kompatibilní s některou z 24 jehličkových tiskáren. Rozlišení je tedy také obdobné 24 jehličkové tiskárně a je asi 360 DPI (14,2 bodu/mm). Velikou výhodou je stabilita odstínu

tisku a oddělitelnost jednotlivých kapiček. Tyto vlastnosti jsou v poslední době využívány k tisku pomocí několika barevných inkoustů, realizujících barevný tisk těmi nejjednoduššími prostředky. Protože se tyto tiskárny vyznačují technologickou nenáročností a také malými rozměry, jsou často využívány jako příruční přenosné tiskárny k počítačům typu laptop a notebook. Jedinou současnou nevýhodou je relativně vysoká cena inkoustu, avšak vzhledem k tomu, že tato technologie je relativně nová, je možné předpokládat velké rozšíření těchto nenáročných tiskáren a tím i snížení provozních nákladů na tisk jednotlivé stránky.

Jako novinka se v poslední době objevily faxy firmy Panasonic, které místo tisku na teplocitlivý papír používají normální inkoustový tisk. Takže to vypadá, že se ledy konečně hnuly a třeba přijde jednou doba, kdy budeme mít doma u ZXS
HP DeskJet695C všichni.

Laserové tiskárny

Tyto tiskárny jsou prvním typem tiskáren, které tisknou dokument po stránkách a ne pouze po řádkách jako tiskárny předchozích typů. Z tohoto způsobu tisku vyplývají určité vlastnosti, charakteristické právě pro stránkový tisk. Jednou z nejvýznačnějších je tvorba celé tiskové strany najednou v paměti tiskárny. Protože je celá tisková strana tvořena v pracovní paměti tiskárny najednou, je nutné, aby tato paměť měla dostatečnou kapacitu. Z tohoto důvodu jsou minimální konfigurace laserových tiskáren nabízeny s 512 K pracovní paměti. Pro tvorbu složitých obrazových objektů však bývá často využíváno 5 MB a více rozšířené pracovní paměti. Další zvláštností je možnost použití zvláštního jazyka pro popis stránky tisku, kterým je postupně tvořena celá tisková plocha najednou. Také lze často využít popisu strany pomocí matematicky definovaných grafických objektů, podobně jako u kreslících strojů, plotterů. Kromě těchto speciálních možností popisu stránky se do popředí zájmu výrobců a uživatelů laserových tiskáren dostává specializovaný jazyk popisu tiskové stránky, jazyk PostScript. Tento jazyk umožňuje popis objektů prostředky vyššího programovacího jazyka a tím i prakticky provést realizaci popisu stránky tisku nezávisle na výstupním tiskovém zařízení.

Laserové tiskárny jsou v současnosti schopné tisknout na mnoho různých typů potiskovaných materiálů. Nejčastěji se používá xerografický papír, který má nejvhodnější vlastnosti podporující tuto technologii tisku. Tisk se provádí pomocí nanášení práškového toneru, který je v dalším kroku zpracování listu v tiskárně vytvrzován. Reálná rozlišovací schopnost současných laserových tiskáren je okolo 1200 DPI (47,2 bodu/mm), nejvíce je však využíváno rozlišení okolo 300 DPI (11,8 bodu/mm). Také se v současnosti začíná rychle rozvíjet výroba laserových tiskáren umožňujících tisk barevných dokumentů.

Laserové tiskárny jsou velmi rychlé při tisku (okolo 4-8 stran za minutu), tiché při provozu a přitom je kvalita výsledných dokumentů velice dobrá. Nevýhoda laserových tiskáren spočívá v jejich ceně, jak pořizovací, tak i ceně souvi-sející s náklady na provoz. V současné době je jejich pořizovací cena kolem Kč 12.000,-- bez DPH.

Osvitové tiskové přístroje

Tyto tiskové stroje představují vrchol tiskových technologií. Jejich výstupem jsou filmové fólie s nasvětlenými tiskovými objekty pro další (ofsetové a hlubotiskové) zpracování. Rozlišení u těchto tiskových strojů často přesahuje tisíce bodů na palec a to znamená, že toto rozlišení se vyrovná nebo i přesahuje rozlišení klasické fotografie. Tyto stroje většinou produkují pouze fólie s tiskem separovaných barev z výsledných objektů pro barevné zpracování ve velkých tiskárenských podnicích. Jejich cena odpovídá dosahované kvalitě tisku, a proto jsou pro průměrného uživatele zcela nedostupné. Jako standardní rozhraní je použit jazyk PostScript. Často bývá předzpracování informací pro tisk prováděno ve speciálních strojích, tzv. RIP (Raster Image Processor), které provedou přepočet popisu tiskové strany v jazyce PostScript na bitovou mapu výsledného tisku a tata mapa je potom předána osvitové jednotce k nasvícení filmové fólie.

Myslím si, že pro získání obecného přehledu o tiskárnách by to mohlo stačit. Příště se podíváme na kompletní popis řídících příkazů maticových (jehličkových) tiskáren.

Na shledanou se těší

-AIDS-

DATA GEAR

Data Transfer Accelerator
Graphic Environment Adaptor Racer

Mnozí z pravidelných čtenářů YS jistě velmi dobře vědí, co je to DataGear. Mohli jste se o něm dočíst v dřívějších číslech našeho časopisu. Schéma, které Vám nyní předkládáme, není úplně shodné s původním návrhem, ale je plně funkční a dostačující. Takže, DataGear ještě jednou.

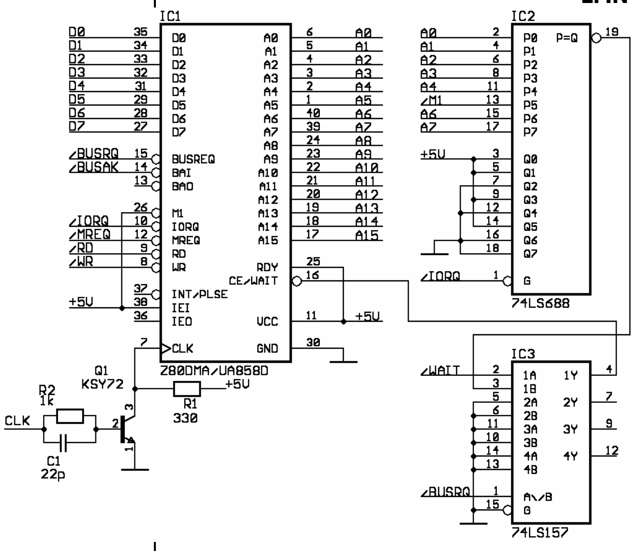
DataGear je periferní zařízení připojitelné k počítačům ZXS, které slouží k rychlým operacím s daty. Umožňuje rychlý přesun dat v paměti, mezi pamětí a porty a mezi porty. Na obrázku vidíte elektrické schéma. Zařízení se skládá pouze ze tří integrovaných obvodů (IO). Srdcem je integrovaný obvod IC1 Z80-DMA (Direct Memory Access). Velkou výhodou tohoto obvodu je, že ke své činnosti vystačí pouze s jedním portem. V našem zapojení je použit port 11 (BIN 00001011). Jako dekodér tohoto portu je použit obvod IC2. Obvod IC3 je nutný pro připojení DataGearu k počítačům ZXS +2A, ZXS +2B a ZXS +3. Zajišťuje multiplexaci vstupního signálu CE/WAIT obvodu DMA. Pokud budete DataGear používat pouze na počítačích ZXS 48, ZXS 48+, ZXS 128+ a ZXS +2, můžete obvod IC3 úplně vynechat a spojit pin 19 obvodu IC2 s pinem 16 obvodu IC1. Takto se vám počet integrovaných obvodů sníží na pouhé dva. Při konstrukci DataGearu dodržte přesně hodnoty součástek. Obvod DMA musí být připojen na sběrnici počítače co nejkratšími spoji.

Při navrhování MB-02+ jsme narazili na zajimavý problém. Ten se týká obvodu Z80-DMA. Zjistili jsme, že pro zapojení se ZXS nelze používat většinu vyráběných typů obvodu Z80-DMA. Konkrétně jsme zkoušeli DMA od těchto výrobců: Zilog, Mostek a Texas Instruments. Ani jeden není vhodný. Nepoužitelnost se projevuje v tom, že nefunguje jeden z typů přenosu, a to přenos paměť-port, což je způsobeno nestandardním časováním signálu /IORQ. Jediný použitelný typ je zatím UA858D vyráběný firmou RFT, která ovšem již neexistuje, protože byla z bývalé NDR. Tento obvod funguje absolutně bez problémů.

Vlastnosti:
- vyzkoušeno se ZXS 48, ZXS 48+/Delta, ZXS 128, ZXS +2, ZXS +2A, ZXS +3
- plně kompatibilní s DMA v MB-02+
- rychlost přenosu na ZXS 128 až 865 KB/s

Tímto článkem bychom chtěli otevřít takový hardwarový miniseriál, který se samozřejmě bude týkat věcí kolem ZXS. Takže již teď se můžete těšit na ULA-Corrector (tzv. "Deprchátor"-pozn. editora), 512KB SRAM, NMI, HDD-Controller atd. Pokud máte nějaké zajímavé zapojení, pošlete nám ho a my jej velmi rádi otiskneme.
Pěkné zážitky se Z80-DMA vám přeje 8BitCompany Laboratories.
DataGear je ochranná známka firmy 8BitCompany. Toto zařízení nesmí být vyráběno ani prodáváno za účelem zisku.

-LMN-



Tečka.

Nová verze Bible, co se šušká a co huláká, co se děje a neděje... to vše najdete v Tečce!

Genesis

verzia 1.5

1. Na počiatku stvoril Clive Sinclair plošný spoj s medenými cestičkami.
2. A medené cestičky boli neladné a pusté a práznota sa vznášala nad nimi.
3. A Clive riekol: Nech sú odpory a kondenzátory! A boli odpory a kondenzátory.
4. A Clive videl odpory a kondenzátory, že sú funkčné, a Clive oddelil odpory a kondenzátory na jednu stranu plošného spoja.
5. A Clive nazval stranu s odpormi a kondenzátormi "strana súčiastok" a druhú stranu s prázdnotou nazval "strana spojov". A bola kladná polperióda, a bola záporná polperióda, prvý hodinový takt.
6. A Clive riekol: Nech je prekážka medzi vodivými medenými cestičkami a súčiastkami a nech elektricky delí medené cestičky od súčiastok!
7. A Clive učinil prekážku a elektricky oddelil medené cestičky, ktoré sú na plošnom spoji, od súčiastok, ktoré sú nad plošným spojom. A bolo tak.
8. A Clive nazval prekážku izolačným lakom. A bola kladná polperióda, a bola záporná polperióda, druhý hodinový takt.
9. A Clive riekol: Nech sa zhromaždia kontakty na strane súčiastok na jedno miesto a nech sa ukáže kontaktové pole! A bolo tak.
10. A Clive nazval kontaktové pole klávesnicou a spoje kde kontakty neboli nazval prívodmi ku klávesnici. A Clive videl, že je to dobré.
11. A Clive riekol: Nech vydá plošny spoj rýchle integrované obvody, dekodéry dekódujúce signály, logické členy vykonávajúce logické funkcie podľa svojho druhu v ktorých budu ich výsledky, na plošnom spoji. A bolo tak.
12. A plošný spoj vydal rýchle integrované obvody, dekodéry dekódujúce signály, logické členy vykonávajúce logické funkcie podla svojho druhu. A Clive videl, že je to dobré.
13. A bola kladná polperióda, a bola záporná polperióda, tretí hodinový takt.
14. A Clive riekol: Nech sú taktovacie kryštály na plošnom spoji, aby delili kladnú polperiódu od zápornej, a budú na znamenia, na určité takty a na pedesiatiny sekundy.
15. A budú časovacími generátormi na plošnom spoji aby časovali ostatné súčiastky. A bolo tak.
16. A Clive učinil dva kryštály, 14 MHz kryštál, aby panoval nad procesorom, a 4.43 MHz kryštál, aby panoval nad farebným modulátorom PAL.
17. A Clive ich dal na plošný spoj, aby časovali ostatné súčiastky
18. a aby generovali pracovné frekvencie procesoru a farebnému modulátoru PAL a aby delili kladnú polperiódu od zápornej. A Clive videl, že je to dobré.
19. A bola kladná polperióda, a bola záporná polperióda, štvrtý hodinový takt.
20. A Clive riekol: Nech sa hemží klávesnica klávesami a nech elektróny lietajú po medenych cestičkach, na tvári plošného spoja.
21. A Clive stvoril tie klávesy, stláčajúce sa, ktorými sa hemží klávesnica, podľa ich druhu, a všeliaké elektróny okrídlene podľa ich druhu. A Clive videl, že je to dobré.
22. A Clive ich požehnal a riekol: Ploďte sa a množte sa a naplnte klávesnicu, a elektróny nech sa množia v medených cestičkách!
23. A bola kladná polperióda, a bola záporná polperióda, piaty hodinový takt.
24. A Clive riekol: Nech vydá plošný spoj najhlavnejšie komponenty: procesor Z80 a videoprocesor ULA podľa ich druhu! A bolo tak.
25. A Clive učinil Zet-osemdesiatku a Ulu. A Clive videl, že je to dobré.
26. A Clive riekol: Učiňme programátora na svoj obraz a podľa svojej podoby, a nech vládne nad klávesnicou a nad všetkými súčiastkami na plošnom spoji.
27. A Clive stvoril programátora na svoj obraz, na obraz Cliva Sinclaira ho stvoril, kódera, hudobníka a grafika ich stvoril.
28. A Clive ich požehnal a Clive im riekol: Vládnite nad klávesnicou a nad každou súčiastkou, ktorá je na plošnom spoji,

programujte, kódujte, komponujte,kreslite a podmaňte si svet zaujímavými a hodnotnými programami.
29. A Clive riekol: Hľa, dal som vám kazdu súčiastku, odpor, kondenzator, dekóder, logický člen, Z80, ULA, každý integrovaný obvod. To všetko vám bude za hardware - nástroj programátora.
30. A procesoru Z80 a videoprocesoru ULA dal som na podporu ich činnosti všetky ostatné súčiastky.
31. A Clive videl všetko, čo učinil, a hľa, bolo to velmi dobré.
A bola kladná polperióda, a bola záporná polperióda, šiesty hodinový takt.

---

1. A dokonané bolo ZX Spectrum i celý jeho hardware i všetko jeho programátorské vojsko.
2. A Clive dokonal siedmeho hodinového taktu svoje dielo, ktore činil, a odpočíval siedmeho taktu od všetkého svojho diela, ktore učinil.
3. A Clive požehnal siedmy takt a posvätil ho, lebo v ňom si odpočinul od všetkého svojho diela, ktore stvoril Clive činiac.
4. To sú rody ZX Spektra a jeho programátora, v pedesiatine sekundy, ked boli stvorené, v pedesiatine sekundy, v ktorej činil Clive Sinclair ZX Spektrum i programátora.

Špeciálne pre ... (všetkých!) napísal

-BUSY-

NeverMore!

Jeden můj kámoš z dětství, tehdy zanícený Spectrista, se neustále chvástal, co dělá nového na ZXS-tu nový efektní loader, tu novou 3D hru, tu perfektní textovku. Nikdy jsem od něho nic neviděl, až na jedno-výpis fragmentu jeho "loaderu" na kusu papíru. Všiml jsem si instrukce XOR C ve zvláštní souvislosti a tak se ptám-co to má dělat? Dostal jsem briskní odpověď-vynuluje to céčko...
Jmenoval se Mira.

-BLS-

Zilog: beze mne by jste se váleli v hnoji!

BLS zažil na loňském DOXYCONu několik šoků v jednom dni: poprvé, když zjistil, že jeho programovatelný kalkulátor Texas Instruments TI-85 (mimochodem opravdu špičková kalkulačka) má v sobě mikroprocesor Zilog Z80 taktovaný na 2 MHz, podruhé při zjištění, že je možno jej programovat na úrovni strojového kódu a napsat tak naprosto fantastické aplikace. Nutno podotknout, že celou dobu (ještě před těmito zjištěními) při používání TI-85 vykřikoval, že ta kalkulačka je tak rychlá, že je v ní určitě zetosmdesátka.
Třetí šok se dostavil k večeru, když DRON přišel s otázkou, jestli BLS neví, jak naemulovat paměťovou cartridge s hrami pro Nintendo GameBoy. Pakliže se ptáte jako BLS k čemu to, tak vězte, že DRON si chtěl pouze vykuchat pouky do svých oblíbených her. On totiž i ten GameBoy má ve svých útrobách... hádejte co! A tak vyvstává otázka, zdali se někomu nechce naprogramovat (lépe řečeno převést, naprogramováno již je) emulátor TI-85 nebo GameBoye na ZXS...



BLS a LMN z 8BC (vlevo a vpravo) za YS se přátelsky objímají s vydavatelem ZXM MATSOFTem.

Bombónek na konec!

Vytáčejí vás mediálně exponovaní lidé? Tak jim to nandejte a zavolejte jim! Abyste ovšem věděli kam, musíte znát telefonní číslo. To zjistíte spuštěním následujícího programu:

```
1 RESTORE: PRINT "Daniel Hulka:":
FOR x=1 TO 11: READ y: PRINT CHR$
PEEK y;: NEXT x
2 DATA 552,719,860,1547,3188,6126,
6286,9033,9040,9114,9344
```

-8BC-

PC sucks!!!

SFS-01

Spectrum File System

Úvod

V poslednom čase sa veľmi rozvírila hladina okolo pripojenia harddiskov k ZX Spektru. Dokonca už v lete na DoxyCone 98 a potom v zime na ZlinCone 98 Tritol prezentoval prvé rozhranie pre ZX Spektrum umožňujúce pripojenie a ovládanie IDE harddisku. Preto už v lete, na DoxyCone 98 sa udial historický okamih - vznikla KpSH - Komisia pre Štandardizáciu Harddisku. Členmi novovzniknutej komisie vtedy boli Tritol, Pvl, LMN of 8BC, BLS of 8BC, Glip a ja (Busy). Neskôr sa k nej pridali ešte Noro, WWW a Baze. Komisia si uložila za úlohu zaviesť akýsi štandart v používaní harddisku na ZX Spektre a tým zabezpečiť kompatibilitu medzi jednotlivými užívateľmi harddisku. Všetci iste dobre poznáte dnešnú situáciu disketových jednotiek na Spektre - existuje ich niekoľko vzájomne nekompatibilných typov. Hlavným cieľom tejto komisie je zabezpečiť, aby sa tento stav neopakoval aj v prípade harddiskov a aby nevzniklo niekoľko rôznych pripojení a spôsobov používania harddisku na Spektre. Prvým krokom k dosiahnutiu tohto cieľa je niečo, čo sa nazýva Speccy File System.

Čo je Speccy File System?

Speccy File System (ďalej len SFS) je norma, ktorej hlavným cieľom je definovanie spôsobu ukladania dát na harddisku, alebo inými slovami povedané, ako má byť harddisk naformátovaný. V tom sa skrývaju tieto dve veci:

- rozdelenie harddisku na partície
- štruktúra údajov v jednej partícii

Ďalším cieľom sú rôzne doporučenia pre operačný systém, ktoré popisujú ako sa má s takto definovaným formátom harddisku efektívne pracovať.

Samozrejme SFS definuje ukladanie dát nielen na harddisk, ale vo všeobecnosti aj na ľubovoľné iné záznamové médium (ďalej len médium) s blokovým prístupom k dátam (t. j. dáta sú uložené v sektoroch).

Rozdelenie harddisku na partície

Jedným z uznesení komisie bolo, že harddisk bude rozdelený na partície presne tak ako definuje MS-DOS 5.0 a vyšší. Prináša to so sebou niekoľko veľmi pozitívnych dôsledkov: na harddisku je možné si vytvoriť niekoľko partícií, jedna bude klasická MS-DOS, ďalšia bude napríklad Linux ext2, a čo je najhlavnejšie, jedna bude naformátovaná podľa normy SFS. Umožní to veľmi jednoduché používanie toho istého harddisku na PeCi pod MS-DOSom, pod Linuxom, ale zároveň aj na Spektre.

Nízkokapacitné médiá (diskety, ramdisky) nie sú rozdelené, t. j. ako keby celé médium vytváralo jednu partíciu.

Ďalší text je už popis štruktúry údajov v jednej partícii, ktorá je naformátovaná podľa normy SFS.

Mimochodom, čítali ste VIX-ovu básničku "Partícia moja krásna"? Veľmi pekná básnička!

Základné charakteristiky a ohraničenia

Aj keď komisia v kútiku duše dúfala, že sa jej podarí navrhnúť SFS ktorý by nemal absolútne žiadne ohraničenia (pod heslom "Na Spektre je všetko možné!"), predsa len sa napokon radšej rozhodla viac sa držať pri zemi a navrhnúť niečo, čo by bolo rozumným kompromisom medzi nekonečnými možnosťami adresácie diskových kapacít a medzi efektívnou a nie príliš zložitou prácou operačného systému, ktorý bude pracovať nad týmto formátom. Navrhnutý SFS má takého ohraničenia:

Parameter	FAT16	FAT32
Maximálna veľkosť fyzického sektora (bytes)	2^14	2^30
Maximálna veľkosť logického sektora (bytes)	2^14	2^30
Maximálna veľkosť jedného súboru (bytes)	2^32-1	2^32-1
Maximálna kapacita logického média (clusters)	2^14-256	2^30-256
Maximálny počet adresárov na médiu	2^32	2^32
Maximálny počet súborov v adresári	2^32	2^32

Všetky tieto ohraničenia, snáď až na dĺžku súboru, sú zatiaľ viac-menej teoretické, pretože v praxi je hlavným (a jediným) ohraničením fyzická kapacita média. Ale vzhľadom na dnešný rýchly vývoj kapacít harddiskov je dosť dobre možné, že bude k dispozícii za rozumnú cenu harddisk takej kapacity, ktorá spôsobí prechod nejakého vyššie uvedeného ohraničenia z roviny teoretickej do roviny praktickej...

Fyzické a logické sektory

Určite ste si v predchádzajúcej časti všimli pojmy "logický sektor", "fyzický sektor", "cluster". Čo tieto pojmy vlastne znamenajú ?

Fyzický sektor je najmenší balík dát, ktorý môžeme poslať do alebo načítať z harddisku. Obvykle má veľkosť 512 bajtov.

Logický sektor, zvaný tiež cluster, je určité množstvo informácii, s ktorým naraz pracuje operačný systém pri práci s pamäťovým médiom. Jeho veľkosť obvykle býva zhodná s fyzickým sektorom, ale v prípade, že fyzický sektor je príliš malý, logický sektor býva zložený z určitého malého množstva fyzických sektorov.

V tomto popise SFS sa logický sektor označuje priamo pojmom "sektor". V prípade, že sa v popise jedná o fyzický sektor média (v popise fyzických informácii o disku), vtedy je výslovne uvedené že ide o fyzický sektor.

Tak, a dosť bolo úvodných kecov a teraz sa už poďme naostro venovať popisu samotného SFS.

Boot sektor

Boot sektor je ako jediný uložený na pevnej pozícii - v prípade ak logické médium je priamo nejaký fyzický disk (disketa), je uložený hneď na úplne prvom sektore daného média - prvý fyzický sektor na nultej stope a nultej strane. Ak je logické médium jedna z partícií harddisku, potom boot sektor je prvý sektor tejto partície. Vo všeobecnosti je to vždy sektor, ktorý má logické číslo 0.

Boot sektor je veľmi dôležitý, pretože keď operačný systém ide pracovať s médiom, ktoré ešte nemá "nakešované", je boot sektor hneď to prvé, čo operačný systém načíta do pamäti. Slúži teda na počiatočnú identifikáciu média.

Boot sektor má ešte väčší význam pri tzv. bootovatelných médiách. To sú také média, na ktorých je uložený operačný systém, ktorý sa po zapnutí počítača loadne do pamäti, kde sa rezidentne usídli a zabezpečuje všetky operácie s médiom (a mnohé ďalšie veci). V tom prípade sa v boot sektore nachádza tzv. systémový zavádzač, to je taký malý programček ktorý loadne do pamäti operačný systém.

Nezávisle od toho, či je médium bootovatelné alebo nie, prvých 64 bajtov boot sektora obsahuje veľmi dôležité informácie o médiu ako takom:

Identifikácia média

#00-#01	skok na systémový zavádzač (#18,#3E alebo #18,#7E)
#02-#07	"SFS-01" identifikácia formátu
#08-#0F	identifikácia média (8 RND bajtov, ich xor je #55)

Fyzické informácie o médiu

#10-#11	počet stôp alebo cylindrov
#12	počet fyzických sektorov na stopu
#13	počet hláv alebo povrchov
#14	(*) veľkosť fyzického sektora v bajtoch

Logické špecifikácie

#15	(*) veľkosť logického sektora (clustera) vo fyzických sektoroch
#16	(*) hraničný sektor medzi zónou adresárov a zónou dát
#17	maximálna dĺžka pre ukladanie súborov priamo do adresára (x30)
#18-#1B	logické číslo prvého sektora prvej fatky
#1C-#1F	logické číslo prvého sektora druhej fatky

Niečo ako adresárová položka média

#20	atribúty média systémové (napevno nastavené formátovačom)
#21	atribúty média užívateľské (voľne meniteľné užívateľom)
#22-#2B	meno logického média (label)
#2C-#2F	celkový počet sektorov na médiu
#30-#33	dĺžka fatky v sektoroch
#34-#37	dátum a čas formátovania
#38-#3B	dĺžka hlavného adresára v bajtoch
#3C-#3F	logické číslo prvého sektora hlavného adresára

Prídavné identifikácie média (nepovinné, SFS ich len doporučuje)

#40-#5F	identifikácia formátovacej utilitky (napr. F02-V30)
#60-#7F	identifikácia systému ktorý s médiom pracuje (napr. Norix 1.0)

Systémový zavádzač

#80-END	systémový zavádzač, ak médium je bootovatelné

Identifikácia média je osem náhodných bajtov vytvorených pri formátovaní tak, aby po výpočte 1. bajt XOR 2. bajt XOR ... XOR 8. bajt vyšiel výsledok #55. Tieto bajty by mali byť pre každé médium rôzne. Preto kopírovací program, ktorý

by vytváral presnú kópiu média sektor po sektore, by ich mal náhodne zmeniť. Pri práci s médiom podľa nich systém pozná, či má dané médium už "nakešované", alebo nie.

Fatka je kvôli bezpečnosti bežných magnetických médií zdvojená. V prípade, že fatku netreba zdvojovať (napr. ramdisk), začiatok druhej fatky a príslušný bit v systémových atribútoch média je nulový.

Údaje v bajtoch #14-#16 (označené hviezdičkou) sú vyjadrené pomocou N-tej mocniny dvojky, t. j. ak číslo uložené v ľubovoľnom z týchto bajtov je N potom skutočná hodnota, ktorú bajt reprezentuje bude 2^N. Napríklad ak v bajte #15 (veľkosť logického sektora) bude číslo 4 tak potom skutočná veľkosť logického sektora bude 2^4=16 fyzických sektorov.

Pridavné identifikácie média (formátovač a systém) majú z hľadiska definície SFS len čisto informatívny charakter a sú nepovinné. Sú určené len pre užívateľa, ktorý si nechá zobraziť boot sektor v ASCII forme.

Systémové atribúty média

bit 7	použitý systém FAT: 0=FAT16, 1=FAT32
bit 6	zdvojená FAT: 0=nie, 1=áno
bit 5-0	rezervované

Užívateľské atribúty média

bit 7	podpora bitu OPEN v užívateľských atribútoch: 0=nie, 1=áno
bit 6	spôsob alokácie všetkých sektorov: 0=v poradí, 1=čo najnižší
bit 5	spôsob alokácie adresár. sektorov: 0=hneď za, 1=jeden vynechať
bit 4	spôsob ukladania adresárových a dátových sektorov: 0=pomiešané, 1=oddelené
bit 3	ukladanie krátkych súborov priamo do adresára: 0=nie, 1=áno
bit 2-0	rezervované

Spôsob alokácie všetkých sektorov určuje stratégiu alokácie nového sektora pre súbory aj adresáre. Jednotková hodnota bitu znamená, že nový alokovaný sektor by mal mať čo najnižšie logické číslo (mal by byť čo najbližšie k začiatku média). Tento spôsob alokovania zvyšuje spoľahlivosť záznamu na magnetických diskových médiách, ktoré sú známe tým, že záznam do sektorov na okraji disku je vo všeobecnosti spoľahlivejší. Nulová hodnota bitu znamená že alokovať by sa mal ten sektor, ktorý predtým ešte nebol alokovaný, alebo ak také nie sú, tak bol predtým alokovaný čo najdávnejšie a teda najdlhšie zo všetkých bol zmazaný. V praxi to znamená že sektory sa alokujú postupne od začiatku média až do konca bez ohľadu či medzitým niekde vznikli nejaké zmazané sektory. Táto stratégia je vhodná pre médiá ktoré majú rovnakú dobu prístupu ku všetkým sektorom (napríklad ramdisk). Táto stratégia tiež výrazne zvyšuje pravdepodobnosť spoľahlivého obnovenia zmazaného súboru.

Spôsob alokácie adresárových sektorov určuje kde na médiu sa má naalokovať ďalší sektor adresára. Ak je tento bit nulový, potom ďalší sektor by sa mal nachádzať hneď za posledným už existujúcim sektorom, takže sektory adresára by mali ísť bezprostredne za sebou (ak zanedbáme fragmentáciu). Ak je tento bit jednotkový, potom nový sektor sa vytvorí tak, aby medzi ním a posledným už existujúcim bol aspoň jeden voľný sektor. Tento druhý spôsob je vhodný pre urýchlenie prehľadávania adresára na disketách, pretože po loadnutí sektoru adresára do pamäti procesor potrebuje chvíľku času na prehľadanie tohto sektora a v prípade že by ďalší sektor šiel hneď za práve prehľadávaným tak by ho už procesor nestihol naloadovať v jednej otáčke diskety a musel by čakať na ďalšiu otáčku. Druhý spôsob je vhodnejší pre harddisky, ktoré už sami na sebe majú cache pamäť pre niekoľko sektorov, pretože tieto harddisky pri požiadavke o sektor X si do cache loadnú aj niekoľko ďalších sektorov, ktoré má procesor hneď k dispozícii. Na médiách s rovnakou dobou prístupu ku všetkým sektorom nastavenie tohto bitu nemá na rýchlosť žiadny vplyv, ale vtedy SFS odporúča tento bit nechať nulový aby sa minimalizovala fragmentácia adresárov.

Spôsob ukladania adresárových a dátových sektorov určuje či majú byť adresárové sektory s datovými ukladané spolu pomiešané (hodnota bitu=0) alebo oddelene (hodnota bitu=1). V bajte #17 je definovaný hraničný sektor medzi zónou adresárov a zónou dát. Zóna adresárov je časť média prednostne určená na ukladanie adresárových sektorov. Je umiestnená od začiatku média až po číslo sektora definované v bajte #17. Od tohto čísla sektora (včítane) začína zóna dát, ktorá potom pokračuje až do konca média. Ak je bit 1 v atribútoch média

nastavený na jednotku, operačný systém by mal novovznikajúce adresáre umiestňovať do zóny adresárov a podobne dátové sektory súborov do zóny dát. Až v prípade, ak sa daná zóna zaplní, sa sektor určený na uloženie do tejto zóny môže uložiť do druhej zóny. Tento spôsob je veľmi vhodný pre pracovné médiá, na ktorých sa často erasuje, sejvuje a vypisujú sa katalógy. Okrem výpisu katalógu to hlavne optimalizuje aj hľadanie nejakého súboru v adresároch a vôbec prácu s adresármi, pretože adresárové sektory sú sústredene na začiatku média a nie je potrebné kvôli nim príliš veľa seekovať. Na médiách s rovnakou dobou prístupu ku všetkým sektorom tento bit neprináša žiadne zrýchlenie práce. V prípade, že bit 1 v atribútoch média je nulový, operačný systém ukladá všetky nové sektory na médium rad za radom bez ohľadu na to či je to adresár alebo dátový sektor. Tento druhý spôsob je výhodný vtedy, ak potrebujeme mať na disku menšie množstvo menších súborov. V kombinácii so spôsobom alokácie sektorov čo najbližšie k začiatku média (pozri ďalej) tento spôsob optimalizuje seekovanie diskového média pri nahrávaní súborov na najnižšiu možnú mieru.

Ukladanie krátkych súborov priamo do adresára slúži na minimalizáciu nevyužiteľného miesta na médiu, ak veľkosť súboru je oveľa nižšia ako veľkosť alokačnej jednotky - logického sektora. V prípade, že je tento bit nastavený na jednotku, a ak dĺžka práve ukladaného súboru nepresiahne číslo 30 krát hodnota z bajtu #17, tak data súboru sa neuložia do samostatného dátového sektora, ale za adresárovou položkou súboru sa do adresára ešte pridajú položky typu rozšírenie, do ktorých sa uložia data súboru. Potrebný počet týchto rozširujúcich položiek je daný dĺžkou súboru, maximálny počet je daný bajtom #17. Možnosť uchovávať krátke súbory je veľmi vhodná pre rozsiahle archivačné médiá, ktoré majú veľmi veľké sektory a kde by nejaký povedzme stobajtový zavádzací basic zbytočne zaberal celý jeden sektor média. Avšak v prípade, že sa požaduje veľká rýchlosť práce s adresárom, nie príliš komplikovaný sekvenčný zápis do súboru a kde nie je až tak kritické efektívne využitie voľného miesta na médiu, je efektívnejšie ukladať aj krátke súbory do samostatných dátových sektorov.

FAT - file alocation table

Fatka (alebo ľudovo povedané "tučná tabuľka") je pravdepodobne najdôležitejšia tabuľka na celom médiu. Definuje sa v nej, ktoré sektory sú voľné a ktoré sú použité a pre ktorý súbor, v akom poradí idú sektory súboru za sebou, definujú sa tam rezervované sektory pre špeciálne účely, a tiež sú v nej zakódované aj všetky dĺžky súborov. Najdôležitejšia informácia je práve informácia o tom, ako za sebou idú sektory jednotlivých súborov.

Na médiu môže byť jedna alebo dve fatky. V prípade, že na médiu je len jedna fatka, logické číslo prvého sektora druhej fatky je nulové a ták isto je nulový aj bit 6 v identifikačnom bajte média (offset #20). Každá fatka je umiestnená na médiu kvôli efektívnemu random prístupu v sektoroch ktoré idú bezprostredne za sebou od prvého sektora, ktorého číslo je definované v boot sektore. Tento prístup umožnuje mať fatku umiestnenú kdekoľvek na médiu (hoci aj kvôli efektívnemu seekovaniu v strede média) avšak fatka musí ostať celá pokope, nemôže byť rôzne rozptýlena po médiu.

Fatka sa skladá z položiek, každá položka zodpovedá jednému sektoru média. Položky zaberajú podľa potreby 16 alebo 32 bitov. Podľa toho potom rozlišujeme či sa jedná o systém FAT16 alebo FAT32. Bajty sú v položke usporiadané vždy od najnižšieho po najvyšší. FAT32 sa používa, ak počet sektorov na médiu je večší než umožňuje uadresovať FAT16, čo je 2^14-256 alebo 16128 sektorov, a/alebo ak veľkosť sektora (clustera) je väčšia ako 2^14 alebo 16384 bajtov.

Význam hodnôt v položkách FAT tabuľky	**FAT16**	**FAT32**
	hi.lo - hi.lo	hi.......lo - hi.......lo
Voľný sektor, nepoužitý	00.00	00.00.00.00
Voľný sektor, zmazaný	00.01 - 7F.FF	00.00.00.01 - 7F.FF.F7.FF
Posledný sektor súboru	80.00 - BF.FF	80.00.00.00 - 80.00.3F.FF
		80.00.40.00 - BF.FF.FF.FF
Priebežný sektor súboru alebo smerník na další sektor	C0.00 - FE.FF	C0.00.00.00 - C0.00.3E.FF
		C0.00.3F.00 - FF.FF.FE.FF
Špeciálny sektor (systém)	FF.00 - FF.EF	FF.FF.FF.00 - FF.FF.FF.EF
Chybný sektor	FF.F0 - FF.FE	FF.FF.FF.F0 - FF.FF.FF.FE
Fyzický koniec média	FF.FF	FF.FF.FF.FF

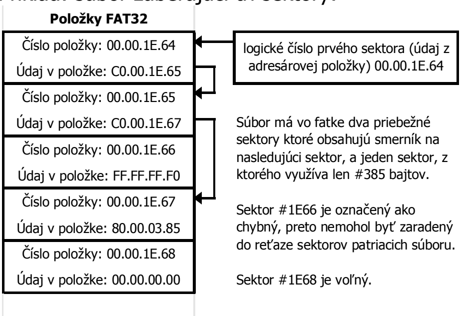
Posledný sektor súboru obsahuje po vynulovaní najvyššieho bitu položky presný počet bajtov ktoré ešte súbor zaberá v tomto sektore. Ak je tento počet nulový, potom skutočný počet týchto bajtov je rovný veľkosti sektora, t. j. sektor je využitý celý => dĺžka súboru je delitelná veľkosťou sektora v bajtoch.

Zmazanie súboru len vynuluje najvyšší bit FAT položky. Všetky položky od 00.00.00.00 do 7F.FF.FF.FF sa chápu ako voľné sektory ktoré sa pri sejve môžu použiť. Špeciálny sektor je napr.

boot, prípadne na bootovatelnom médiu operačný systém, ktorý sa nenachádza v klasickom súbore. Neexistujúci sektor je keď napr. médium má len 250 sektorov, fatka je dlhá 250*4=1000 bajtov ale keďže fatka zaberá celočíselný počet sektorov tak zvyšných 24 bajtov fatky obsahuje hodnotu FF.FF.FF.FF.

Volné sektory pripravené na alokovanie sa od už obsadených alokovaných sektorov jednoznačne líšia najvyšším bitom položky. Volné sektory ho majú nulový, obsadené položky jednotkový. Preto sa pri alokácii ďalších sektorov na médiu môžu vyberať len sektory ktoré majú tento bit nulový.

Príklad: súbor zaberajúci tri sektory:



Doporučenie pre OS

Aby operačný systém nemusel riešiť dva prípady FAT systému (FAT16 a FAT32), tak v prípade dostatočnosti 16 bitovej fatky (médium pod 16128 sektorov) sa FAT16 môže pri spracovaní v OS logicky konvertovať na FAT32 nasledujúcim spôsobom: Medzi bit 13 a 14 16-bitovej hodnoty FAT16 sa vloží nových 16 bitov s hodnotu podľa vyššieho bajtu pôvodnej hodnoty: ak je vyšší bajt rovný 255, potom bity budú jednotkové, ak iný, tak nové bity budú nulové. Tým vznikne nová zodpovedajúca hodnota FAT32. Spätná konverzia je tiež veľmi jednoduchá, predtým pridané bity stačí z hodnoty vynechať.

Štruktúra adresárov a súborov

Hlavným cieľom pamäťového média je uchovávanie a archivovanie súborov. Aby však nemuseli byť všetky súbory na médiu neprehľadne pomiešané na jednom mieste, SFS definuje adresáre, ktoré umožnia mať súbory na médiu rôznym spôsobom roztriedené a tým sa zvyšuje prehľadnosť uloženia súborov a vo väčšine prípadov tiež aj rýchlosť prístupu k súborom.

Všetky adresáre sú na médiu organizované do stromovej štruktúry. Základom tejto štruktúry je hlavný adresár (root). Tento adresár sa volá presne ako logické médium (label) definované v boot sektore, a cez toto meno by malo byť možné do neho pristupovať.

Adresár je postupnosť sektorov presné taká istá ako obyčajný súbor, ale od obyčajného súboru sa líši tým, že jeho dátový obsah je interpretovaný operačným systémom ako postupnosť adresárových položiek. Adresár vždy musí ležať v samostatných sektoroch, nesmie byt umiestnený v rozširujúcich adresárových položkách aj keby jeho celková dĺžka bola menšia ako hranica definovaná v bajte #17 boot sektoru a bit v systémových atribútoch média povoľujúci ukladanie súborov kratších ako táto hranica by bol nastavený. Podobne tak adresáre na rozdiel od súborov nemôžu byť komprimované a zakódované.

Adresárové položky sú vždy 32 bajtové štruktúry, z ktorých sa skladajú jednotlivé adresáre. Každá adresárová položka musí byť jeden z týchto typov:

- súbor — obsahuje informácie o jednom súbore
- adresár — obsahuje informácie o jednom adresári
- linka — odkaz na iný súbor/adresár/linku/komentár
- komentár — textový komentár uložený priamo v adresári
- rozšírenie — rozšírenie predchádzajúcej položky v adresári

Jednotlivé typy položiek sa od seba odlišujú bitmi 4,5,6 v identifikácii položky (pozri ďalej).

Adresárová položka - súbor

Súbory sú vlastne to najdôležitejšie a najhlavnejšie, čo treba mať uložené na médiu. Adresárová položka súboru obsahuje všetky dôležité informácie o súbore. Sú to jednak klasické informácie nachádzajúce sa v hlavičke súboru používanej pri práci s magnetofónom, a tiež aj ostatné informácie vyplývajúce zo štruktúry ukladania dát v SFS a možností, ktoré navyše od magnetofónu poskytuje.

#00	identifikácia položky (11xxssss, systémové atribúty)	
#01	informačný bajt	Klasická magnetofónová hlavička
#02-#0B	10 znakov meno súboru	
#0C-#0D	dĺžka súboru podľa hlavičky	
#0E-#0F	štart riadok, premenná, začiatočná adresa	
#10-#11	dĺžka čistého basicu bez premenných	
#12	atribúty súboru (užívateľské atribúty)	
#13	flagbajt tela	
#14-#17	dátum a čas poslednej modifikácie súboru	
#18-#1B	dĺžka tela (skutočná dĺžka súboru v bajtoch)	
#1C-#1F	umiestnenie súboru na médiu	

Štruktúra položky je navrhnutá tak, aby kvôli kompatibilite s magnetofónom mohla uchovávať aj súbory bez hlavičky. Formát uloženia dátumu a času je zhodný s formátom používaním v MS-DOSe, detailne bude vysvetlený neskôr.

Umiestnenie súboru na médiu špecifikuje kde sú uložené dáta súboru. Ak súbor nemá telo alebo telo má nulovú dĺžku, potom sú bajty #1C-#1F nulové. Ak je súbor umiestnený vo svojich vlastných dátových sektoroch, potom v týchto bajtoch je logické číslo prvého sektora súboru. Ďalšie sektory súboru sú definované vo fatke (v predchádzajúcej kapitole). Ak je však súbor umiestnený v nasledujúcich adresárových položkách, potom tieto bajty majú takúto štruktúru:

#1C-#1D	počet obsadených bajtov - koľko bajtov zaberá súbor v položkách
#1E	počet rozširujúcich položiek použitých na uloženie súboru
#1F	rezervované pre budúce rozšírenie

Ak súbor nie je komprimovaný, potom počet obsadených bajtov je zhodný s dĺžkou tela súboru. Ak je skomprimovaný, jedná sa o počet po komprimácii. Dĺžka je len 16-bitová, pretože súbory dlhšie ako 65536 bajtov sa vždy musia ukladať do vlastných dátových sektorov. Počet rozširujúcich položiek určuje koľko nasledujúcich položiek v adresári sa použilo pre uloženie dát súboru. V každej rozširujúcej položke sa môže nachádzať až 30 bajtov dát súboru. Presná štruktúra rozširujúcej položky je popísaná ďalej.

Informačný bajt #01 okrem štandartne používaných hodnôt

0	program v basicu (program)
1	číselné pole (number array)
2	znakové pole (character array)
3	bližšie neurčený blok bajtov (bytes)

može ešte nadobúdať tieto odporúčané (z hľadiska SFS nepovinné) hodnoty:

4	bezhlavičkový súbor (doporučuje sa kvôli prehľadnosti výpisu)
5	48K snapshot
6	128K snapshot
7	sekvenčný súbor (doporučuje sa kvôli prehľadnosti výpisu)

Ináč hodnoty vyššie ako 3 sú rezervované pre budúce použitie.

Bajt #00 identifikácia položky (systémové atribúty) má takúto štruktúru:

bit 7	platnosť položky: 1=platná, 0=neplatná (napr. zmazaný súbor)
bit 6	1=značka, že adresárová položka definuje súbor
bit 5	umiestnenie súboru v: 0=dátových sektoroch, 1=rozš. položkách
bit 4	hlavička súboru: 0=nie, 1=áno (t. j. 0=bezhlavičkový súbor)
bit 3	kompresia tela súboru: 0=nie, 1=áno
bit 2	kódovanie súboru heslom: 0=nie, 1=áno
bit 1	rezervovaný, mal by byť nulový
bit 0	linearita (sektory súboru idú za sebou): 0=nie/neviem, 1=áno

Linearita je príznak, či dátové sektory súboru a teda aj ich logické čísla idú bezprostredne za sebou. Ak je tento bit jednotkový, potom súbor nie je fragmentovaný. Kompresia a kódovanie tela súboru sú podrobne vysvetlené neskôr v samostantných kapitolách.

Bajt #12, užívatelské atribúty súboru vyzerajú takto:

bit 7	O=Open: 0=všetko v poriadku, 1=súbor nie je uzavretý
bit 6	S=System: 0=normálny súbor, 1=systémový
bit 5	A=Archive: 0=nový súbor, 1=zaarchivovaný

bit 4	H=Hidden: 0=viditeľný, 1=neviditeľný (CAT nevypisuje)
bit 3	D=Deletable: 0=nezmazateľný, 1=zmazateľný (DELETE, ERASE)
bit 2	R=Readable: 0=nečitateľný, 1=čitatelný súbor (pre LOAD)
bit 1	W=Writeable: 0=nemodifikovateľný, 1=modifikovateľný súbor
bit 0	X=eXecutable: 0=nespustiteľný, 1=spustiteľný (NEW,LOAD...)

Nastavený bit OPEN indikuje, že súbor je otvorený pre zápis. To znamená, že v pamäti sa môžu nachádzať dáta, ktoré ešte nie sú sejvnuté na médium. Tento bit sa používa na diagnostické účely - kontrolu že súbor bol korektne uzavretý a teda dáta uložené v ňom sú platné. Vzhľadom na to, že používanie tohto bitu vyžaduje minimálne dva zápisy do položky (jeden pri otvorení súboru a jeden pri jeho uzavretí), je možné si zvoliť, či sa tento bit bude takto využívať. Určené je to bitom 3 v bajte #20 - atribútoch média v boot sektore. Tým sa užívateľ môže rozhodnúť, či chce mať istotu že zápis súboru prebehol v poriadku, alebo radšej dá prednosť pomalšiemu opotrebovávaniu média menším počtom zápisov.

Bit SYSTEM sa odporúča nastaviť pri systémových súboroch na bootovatelnom médiu. Bit ARCHIVE sa odporúča nastaviť, ak k danému súboru niekde existuje záložná kópia (poznámka: v MS-DOSe je to naopak). Tieto dva bity sú len pomocné a z hľadiska SFS nemajú žiadny ďalší význam.

Bit eXecutable indikuje, že súbor je nejaký program, ktorý je možné spustiť. Môže to byť basic, nejaký stroják pre Z80 uložený ako bytes, alebo snapshot.

Ostatné bity majú svoj zvyčajný význam. Vždy hodnota 0 danú činnosť zakazuje a hodnota 1 ju povoľuje. Môže sa zdať, že bit Readable je možno zbytočný (komu by načo už len bol súbor ktorý sa nedá čítať), avšak tento bit je tu kvôli logickej úplnosti definície atribútov súboru.

Adresárová položka - adresár

Každá adresárová položka definujúca adresár obsahuje všetky dôležité informácie práve o tomto adresári na médiu. Poradové číslo položky v nadadresári, v ktorom je uložená, zároveň určuje číslo adresára, cez ktoré je možné do tohto adresára pristupovať.

#00	identifikácia položky (1011xxxx, vzor pre syst. atribúty)
#01	rezervované pre budúce použitie
#02-#0B	10 znakov meno adresára
#0C-#0F	dátum a čas vytvorenia adresára
#10-#11	rezervované pre budúce použitie
#12	atribúty adresára (užívateľské atribúty)
#13	vzor pre užívateľské atribúty vytváraných súborov
#14-#17	dátum a čas poslednej modifikácie
#18-#1B	dĺžka adresára v bajtoch
#1C-#1F	logické číslo prvého sektora

Položka obsahuje dve časové špecifikácie. Prvá určuje kedy bol adresár vytvorený a druhá nesie informáciu o poslednej modifikácii adresára. Modifikácia adresára je akákoľvek zmena v položkách, ktoré sa nachádzajú v adresári.

Vzor pre užívateľské atribúty vytváraných súborov je bajt, ktorý sa vloží do užívateľských atribútov súboru, ktorý bol vytvorený v tomto adresári. Ako defaultnú hodnotu, ktorá sa sem vloží pri vytvorení adresára, sa odporúča používať hodnotu #0F.
Bajt #00 identifikácia položky obsahuje okrem iného aj vzor ako majú vyzerať systémové atribúty pre súbory vytvárané v tomto adresári:

bit 7	platnosť položky: 1=platná, 0=neplatná (napr. zmazaný adresár)	
bit 6	0	značka, že adresárová položka definuje adresár
bit 5	1	
bit 4	1	
bit 3	vzor pre súbory: komprimácia tela súboru: 0=nie, 1=áno	
bit 2	vzor pre súbory: kódovanie súboru heslom: 0=nie, 1=áno	
bit 1	vzor pre súbory: rezervovaný, mal by byť nulový	
bit 0	vzor pre súbory: lineárne ukladanie súboru: 0=nie, 1=áno	

Ak je bit 3=1 tak potom súbory sejvované do adresára by sa mali komprimovať a ak je bit 2=1

tak by sa mali ukladať na médium zakódované heslom. Ak je bit 0=0 tak sektory pre ukladaný súbor sa alokujú normálne podľa stratégie danej bitom 0 v atribútoch média, ale ak je bit 0=1 tak sa síce tiež používa daná stratégia, ale súbor by sa mal uložiť nefragmentovaný, t. j. podľa tejto stratégie sa hľadá prvá voľná, dostatočne veľká medzera voľných sektorov pre uloženie súboru. Ak sa na médiu takáto medzera už nenájde, v tom prípade sa súbor uloží fragmentovane a jeho bit linearity tým pádom ostane nulový.

Bajt #12 užívateľské atribúty adresára:

bit 7	O=Open: 0=všetko v poriadku, 1=adresár nie je uzavretý
bit 6	S=System: 1=adresár obsahuje systémové súbory
bit 5	A=Archive: 1=všetky súbory v adresári sú zaarchivované
bit 4	H=Hidden: 0=viditeľný, 1=neviditeľný (nevypisuje sa)
bit 3	D=Deletable: 0=nezmazateľný, 1=zmazateľný (DELETE, ERASE)
bit 2	R=Readable: 0=nečitateľný, 1=čitateľný súbor (pre CAT)
bit 1	W=Writeable: 0=nemodifikovateľný, 1=modifikovateľný adresár
bit 0	X=eXecutable: 0=nevykonateľný, 1=vykonateľný (prístup k súborom)

Bit OPEN funguje podobné ako pri súboroch. Hodnota 1 znamená že sa operácia zápisu do adresára ešte neskončila a teda dáta môžu byť nekonzistentné. Bit 3 v atribútoch média (boot bajt #20) určuje či sa tento bit má alebo nemá používať.

Bit SYSTEM sa odporúča nastaviť ak adresár obsahuje systémové súbory potrebné pri bootovaní médiu. Bit ARCHIVE sa odporúča nastaviť, ak všetky súbory v adresári majú niekde vytvorenú záložnú kópiu. Podobné ako pri súbore, aj pri adresári sú tieto dva bity len pomocné a z hľadiska SFS nemajú žiadny ďalší iný význam.

Bit Hidden zakazuje zobrazenie adresára pri výpise nadadresára. Na výpis súborov v adresári nemá žiadny vplyv.
Nulový bit Deleteable chráni adresár proti zmazaniu ako celku, avšak už nie proti zmazaniu jednotlivých súborov v ňom. Proti zmazaniu súborov v adresári chráni bit Writeable - ak je nulový, nie je možné adresár akokoľvek modifikovať. Pod modifikáciou adresára sa myslia akékoľvek úpravy adresárových položiek v adresári (zmena mena, atribútov, ... jednotlivých súborov).

Bit Readable povoľuje výpis adresára (CAT). Ak je nulový, je to zhruba asi tak ako keby všetky súbory v adresári mali nastavený Hidden bit.

Bit eXecutable povoľuje daný adresár použiť v ceste k súborom uloženým v tomto adresári. Ľudsky povedané, ak tento bit je nulový, sú súbory v adresári pre datové operácie (read, write, new) neprístupné.

Adresárová položka - komentár

Textový komentár nie je operačným systémom nijak interpretovaný, systém ho iba vypisuje pri výpise adresára. Doporučené použitie je pre rôzne užívateľské poznámky o programoch v danom adresári.

#00	identifikácia položky (1001xxxx, xxxx=rezervované)
#01	počet nasledujúcich rozširujúcich položiek
#02-#1F	30 znakový textový reťazec ako komentár

Do jednej komentárovej položky sa vojde 30 znakový text. Ak je potrebné umiestniť do adresára komentár dlhší ako 30 znakov, možno prvých 30 znakov komentáru umiestniť do tejto položky a zvyšné znaky do potrebného množstva rozširujúcich položiek, ktoré nasledujú bezprostredne za touto položkou. Ak sa komentár vojde do 30 bajtov, bajt #01 je nulový.

Adresárová položka - linka

Linka je odkaz na inú adresárovú položku nachádzajúcu sa v tom istom, alebo prípadne aj inom adresári na médiu. Odkaz na iné médium neumožňuje. Doporučené použitie linky je v prípade, ak je treba určitý súbor mať nahraný vo viacerých adresároch naraz. V tom prípade sa súbor nahrá len do jedného adresára a v ostatných adresároch bude len linka na tento súbor. Alebo v prípade, ak daný súbor má byť viditeľný pod viacerými číslami (menami) v tom istom adresári.

#00	identifikácia položky (1010tttt, tttt=typ linky)
#01	počet nasledujúcich rozširujúcich položiek (ak nejaké treba)
#02-#0B	10 znakov meno súboru (meno linky)
#0C-#1F	20 bajtov špecifikácia cieľa

Bity tttt v identifikácii položky bližšie určujú akým spôsobom linka ukazuje na svoj cieľ. Ak je bit 3=0 potom linka je prvého (jednoduchšieho) typu a vtedy bajty #0C-#1F majú takýto význam:

#0C-#15	10 znakov špecifikácia adresára kde sa nachádza súbor
#16-#1F	10 znakov špecifikácia súboru na ktorý ukazuje linka

Ostatné bity tttt majú takýto význam:

bit 2	vyhodnotenie adresára: 0=absolútne, 1=relatívne
bit 1	špecifikácia adresára: 0=číslom, 1=menom
bit 0	špecifikácia súboru: 0=číslom, 1=menom

Absolútne vyhodnotenie adresára znamená, že daný adresár je alebo sa hľadá medzi podadresármi hlavného adresára. Tento typ linky je možné použiť pre ukazovanie na súbor alebo adresár ktorý sa nachádza v druhej úrovni stromu alebo jeho cesta sa dá vyjadriť v tvare: Disk:\adresár\súbor.

Relatívne vyhodnotenie adresára znamená, že sa východisko cesty neberie od hlavného adresára média ale od nadadresára k aktuálnemu adresáru. Tento typ linky sa dá použiť najlepšie pre ukazovanie na súbor/adresár ktorý sa nachádza na tej istej úrovni stromu ako poloha linky, alebo inými slovami jeho cesta sa dá vyjadriť v tvare: ..\adresár\súbor.

10 znakov špecifikácia može byť 10-znakové meno súboru/adresára alebo 4-bajtové číslo súboru/adresára. V prípade čísla je ďalších 6 bajtov nevyužitých (môže slúžiť na nejakú poznámku). To, či je použité meno alebo číslo, určujú bity 0 a 1 v identifikácii položky.

Druhý (všeobecnejší) typ linky umožňuje definovať cieľ cestou ľubovoľnej dĺžky. Vtedy je tttt bit 3=1. Bajty #0C-#0D v položke vždy určujú dĺžku cesty. Cesta sa skladá z definície jednotlivých adresárov na konci ktorej je adresár alebo súbor podľa toho, na čo ukazuje linka. Počet týchto definícii je zároveň dĺžka cesty. Ak tttt bit 2=0 tak tieto definície sú uložené v osemnástich bajtoch #0E-#1F. Ak by sa cesta nevošla do týchto osemnásť bajtov, je vtedy možné za položku ešte pridať príslušný počet rozširujúcich položiek (do každej sa vojde 30 bajtov tejto cesty). Potom v bajte #01 je zapísaný počet koľko za ňou nasleduje rozširujúcich položiek. Ak stačí 18 bajtov, bajt #01 je nulový (tak isto ako v komentárovej položke).

Ak tttt bit 2=1 potom je táto cesta umiestnená do zvláštneho dátového sektora a štruktúra 20 bajtov špecifikácie cieľa majú nasledovný význam:

#0C-#0D	dĺžka cesty alebo tiež počet definícií
#0E-#13	rezervované (mali by ostať nulové)
#14-#17	dátum a čas poslednej modifikácie linky
#18-#1B	dĺžka dát v bajtoch uložených v dátovom sektore
#1C-#1F	logické číslo prvého dátového sektora linky

Definícia každého adresára alebo cieľového súboru môže byť vyjadrená číslom alebo menom. Číslo môže byť jedno až štvorbajtové.

Štruktúra definície jedného adresára určuje jej prvý bajt.

Špecifikácia menom:
#01..#0A=nasleduje meno, bajt znamená počet znakov mena

Špecifikácia číslom:
#11..#14=nasleduje X-bajtová hodnota

Ostatné hodnoty bajtov sú vyhradené a nemali by byť použité.

Adresárová položka - rozšírenie

Rozšírenie je špeciálny druh položky, v ktorej sa uchovávajú údaje, ktoré sa už nevošli do predchádzajúcej položky. Rozširujúcich položiek môže byť za sebou aj viac (vzniká blok rozširujúcich položiek), avšak nemôžu existovať samostatne, ale len v súčinnosti s nejakými položkami iného typu.

#00	identifikácia položky (1000srrr, srrr=rezervované)

#01 poradie položky v rámci bloku rozšírujúcich poloziek
#02-#1F 30 bajtov dát podľa účelu položky

Bity "srrr" sú rezervované pre budúce použitie, "s" by mal byť nastavený na jednotku a "rrr" by mali ostať nulové.

Do jednej rozširujúcej položky sa vojde 30 bajtov dát. Bajt #01 vždy obsahuje poradie konkrétnej položky v rámci bloku k sebe patriacich rozširujúcich položiek. Napríklad ak nejaká položka potrebuje uložiť v rozširujúcich položkách 80 bajtov dát, čo znamená že je treba použiť tri rozširujúce položky, potom hodnoty bajtu #01 v týchto položkách budú: #01, #02, #03. Podľa týchto čísel operačný systém vie ktoré položky musia ísť bezprostredne za sebou, čo by mal zohľadniť pri rôznych operáciách s adresárom (napr. utrasenie, zoradenie podľa nejakého kritéria) aby nevhodným premiestnením položiek neporušil konzistenciu dát. Pomocou tohto poradového čísla sa ošetruje situácia, keď užívateľ si dá urobiť výpis katalógu od X-tej položky, pričom operačný systém zistí, že X-tá položka je toto rozšírenie s poradovým číslom Y, tak vie, že položka využívajúca blok rozširujúcich položiek má číslo X-Y.

Formát dátumu a času

Formát času a dátumu je prevzatý z MS-DOSu. Vlastnosti formátu:

- jednoduché kódovanie a dekódovanie
- kompletná špecifikácia času zaberá len 4 bajty
- priamym porovnávaním 4-bajtovej hodnoty sa dá určiť čo je mladšie/staršie
- kódovanie časovej špecifikácie od 1. 1. 1980 do 31. 12. 2107
- a pre úplnosť aj jedna nevýhoda: umožňuje zakódovať len párny počet sekúnd

bajt	#17	#16	#15	#14
bit	76543210	76543210	76543210	76543210
údaj	YYYYYYYM	MMMDDDDD	HHHHHMMM	MMMSSSSS

rok mesiac deň hodiny minúty 2x sekundy

Skutočný počet sekúnd dostaneme ak číslo z bitov 0-4 bajtu #14 vynásobíme dvomi. Preto sa tu dá zakódovať len párny počet sekúnd. Skutočný počet rokov dostaneme ak k číslu z bitov 1 až 7 z bajtu #17 pripočítame hodnotu 1980. Všetky ostatné údaje sú interpretovateľné priamo.

Ak sa vo všetkých štyroch bajtoch nachádzajú nuly, potom dátum a čas nie sú definované (napr. daný OS nepodporuje časové špecifikácie). Ak bajty #16 a #17 obsahujú hodnotu #FF tak to znamená že dátum súboru prekročil hodnotu 31. 12. 2107.

Komprimácia súborov

Niekedy je užitočné, aby súbor bol uložený na médiu v skomprimovanom stave, pretože tak zvyčajne zaberá menej miesta. Preto SFS definuje aj komprimáciu súborov. Avšak v prípade požiadavky na uloženie súboru v skomprimovanom tvare sa súbor uloží na médium skomprimovaný len vtedy, ak po komprimácii zaberie menej dátových sektorov ako v neskomprimovanom stave.

V prípade, že súbor je na médiu uložený ako skomprimovaný, jeho skutočná dĺžka v adresárovej položke stále znamená jeho skutočnú dĺžku koľko zaberal povodne neskomprimovaný. Informácia o tom, koľko bajtov súbor zaberá po komprimácii, je uložená vo FATke - je to počet alokovaných sektorov a z posledného sektora počet skutočne využitých bajtov.

Aby sa príliš nespomalilo čítanie a zapisovanie skomprimovaných súborov, pracuje komprimácia na veľmi jednoduchom princípe. Akonáhle sa v súbore nachadzajú aspoň tri rovnaké bajty bezprostredne za sebou, namiesto nich sa do skomprimovaného súboru uloží len ich počet a jeden z nich.

Napríklad máme blok bajtov:

```
11 22 33 33 44 55 8A 8A 8A 8A EE FF
00 00 00 00 00 00 00 00 99 88 77
```

Skomprimovaný blok bude vyzerať takto:

```
06 11 22 33 33 44 55 84 8A 02 EE FF
88 00 03 99 88 77
```

Skomprimovaný blok je rozdelený na úseky, každý úsek začína špeciálnym bajtom (označené bajty):
bit 7=príznak komprimácie
bit 6-0=počítadlo bajtov N, 0=128, 1=129, 2=130 bajtov.

Ak bit 7=0 tak úsek obsahuje N nekomprimovateľných bajtov (t. j. nejdú za sebou aspoň tri rovnaké bajty).

Ak bit 7=1 tak úsek pozostáva z N rovnakých bajtov. V tom prípade je dátový bajt uvedený len raz.

Na to, aby sa pri komprimáci začalo komprimovať, musia za sebou ísť aspoň tri rovnaké bajty. Ak sú len dva, uložia sa ako nekomprimovateľné. Ak je počet rovnakých/ nerovnakých bajtov v bloku za sebou väčší ako 130, tak sa vždy po 130 bajtoch úsek uzavrie a začne sa generovať nový úsek.

Najnepriaznivejsia kombinácia bajtov sú samé nekomprimovateľné bajty. Vtedy sa skomprimovaný blok predĺži o 1/130 svojej dĺžky, pretože po každom 130. bajte je treba uložiť špeciálny bajt ktorý nesie informáciu že daný úsek obsahuje 130 nekomprimovaných bajtov. Vtedy sa samozrejme daný súbor uloží na médium nekomprimovaný.

Poznámka: Podobný typ komprimácie používajú aj kopírovacie programy BS COPY 06, 07, 128=, 128#.

Kódovanie súborov

Podobne ako komprimácia je tiež niekedy užitočné chrániť nejaký konkrétny súbor heslom. Napríklad užívateľ má v súbore uložené nejaké tajné zdrojáky alebo intímne informácie ktoré by sa nemali dostať na verejnosť. SFS preto definuje spôsob, ako súbor zakódovať pomocou nejakého užívateľom zadaného hesla.

Základná myšlienka kódovania je takáto: heslo jednoznačne určuje nejakú pomerne zložitú postupnosť bajtov, ktorou sa "vyxoruje" daný súbor. Dekódovanie súboru je potom analogické: súbor sa vyxoruje touto postupnosťou bajtov ešte raz a znovu je v pôvodnom nezakódovanom tvare.

V prípade, že heslo použité pri dekódovaní súboru nie je presne to isté ako heslo použité na zakódovanie súboru, bude dekódovacia postupnosť bajtov iná ako kódovacia a to, čo z pôvodného súboru vznikne, bude v podstate kopa nepoužiteľných bajtov.
Pri dekódovaní súboru neexistuje žiadny exaktný mechanizmus ktorý by indikoval či súbor je dekódovaný správne, alebo zle, pretože existencia takéhoto mechanizmu by viac či menej uľahčila hľadanie neznámeho hesla.

Algoritmus kódovania a dekódovania

Majme reťazec znakov dĺžky H a súbor dĺžky S:

```
char heslo[H];
char súbor[S];
```

a takýto generátor pseudonahodných čísel G[n]:

```
G[n+1]=(5 * G[n] + 7) MOD 65536
```

potom súbor bude zakódovaný takto:

```
Pre x=0 až S-1 urob:

súbor[x]=súbor[x]  XOR  Hi(G[x])  XOR
Lo(G[x]) XOR heslo[x MOD H];
```

pričom začiatočná hodnota generátora G[0] sa určí takto:

```
high=0; low=0;
Pre i=0 až H-1 urob:
low=(low+heslo[i]) MOD 256;
high=RRCA(high) XOR heslo[i];
G[0]=256*high+low
```

RRCA je operácia totožná s rovnomennou inštrukciou rotácie procesora Z80.

Vzhľadom na to, že perióda generátora pseudonahodnych čísel je 65536, odporúča sa pre lepšie zakódovanie súboru zadávať heslo, ktorého dĺžka nie je súdeliteľná s číslom 65536. Napríklad akékoľvek heslo ktoré má nepárny počet znakov. Najideálnejšie sú heslá ktorých počet znakov je nejaké prvočíslo väčšie ako 7.

Záver

Možno na prvý pohľad vyzerá celá definícia SFS-01 veľmi zložito a komplikovane, ale veľmi veľa vecí z nej sú rôzne špecialitky zabezpečujúce efektívnejšiu prácu operačného systému alebo komfort užívateľa, ktoré jedno-duchšie verzie operačných systémov nemusia využívať. Tiež v niektorých štruktúrach SFS sa nachádzajú redundantné dáta, t. j. dáta, ktoré sa nachádzajú už v iných štruktúrach alebo ktoré je možné z iných dát jednoznačne určiť. Napríklad dĺžka súboru je na médiu uložená trikrát - v hla-vičke súboru, v dĺžke tela a ešte sa aj dá určiť z FATky. Všetky redundantné dáta sú v SFS zámerne kvôli diagnostikovaniu dátovej konzistencie štruktúry údajov a kvôli efektívnejšej práci operačného systému.

-BUSY-